November 13, 1998

PHASE 1 FINAL REPORT:

AN ARTICULATORILY CONSTRAINED, MAXIMUM

LIKELIHOOD APPROACH TO SPEECH RECOGNITION

John Hogden, David Nix, & Patrick Valdez

Los Alamos National Laboratory

For more information, contact the first author at:
MS B265
Los Alamos National Laboratory
Los Alamos, NM 87545
phone: (505) 665-0134
FAX: (505) 665-5220
email: hogden@lanl.gov

Table Of Contents

1. Executive Summary

The task in speech recognition is to be able to speak into a computer microphone and have the computer type out what was said. While speech recognition systems are commercially available for limited domains, state-of-the-art systems have only about a 60%-65% word recognition rate on casual speech, i.e., telephone conversations. Since speaking rates of 200 words per minute are not uncommon in casual speech, a 60% word recognition accuracy implies approximately 80 errors per minute -- an unacceptably high rate for many applications. Furthermore, recognition performance is not improving rapidly. Improvements in word recognition accuracy of a few percent are considered "big" improvements, and recognition rates of the best systems on the Switchboard data have been between 64.9% and 61.2% for three consecutive years, although they have improved from only 52% recognition four years ago.

For various reasons, including poor performance in real world situations, several agencies have been looking for alternatives to the hidden Markov models (HMMs) that are the best current tool for speech recognition -- particularly alternatives that incorporate more knowledge of speech production. Thus, Maximum Likelihood Continuity Mapping (MALCOM), which has been shown to be able to find a mapping between speech acoustics and speech articulator positions (e.g. tongue, jaw and lip positions), while sharing the probabilistic framework that makes HMMs so powerful, is an attractive approach.

An important achievement of the last year was to invent a workable acoustic model capable of being incorporated into the current state-of-the-art speech recognition packages in place of hidden Markov models. This is important in that the acoustic models we created are based on articulation, and so should have a good chance of eventually outperforming HMMs (although they are not yet as good as HMMs).

The word model we devised incorporates a new invention called Multiple-Observable MALCOM (MO-MALCOM). Research funded through this grant showed that MO-MALCOM has the ability to distinguish phonemes better than measured articulator positions. This is an important finding, especially since recent work by Roweis at Caltech demonstrated that articulator data could be used in conjunction with acoustic data to get nearly perfect recognition performance on a speaker-dependent data set (Roweis, 1998).

In fact, the ability of MO-MALCOM to distinguish phonemes is remarkable considering that MO-MALCOM was not using information from word-level models or information about the phonetic context. While we know of no way to directly compare MO-MALCOM's results to HMM results, it seems doubtful that HMM phoneme models that were not incorporated into word or mulit-phone models could perform with comparable accuracy.

The work quantifying MO-MALCOM's ability to discriminate phonemes argues that our current speech recognition system is not limited by the MO-MALCOM word model, but by the various components of the recognition algorithm that were not given much attention by this project, i.e. the word lattices, prior word probabilities, the dictionary, etc. Sophisticated versions of the non-MALCOM components have been implemented by other labs around the country (at considerable expense) and, perhaps, should not be re-invented at Los Alamos National Laboratory.

We tested our speaker-dependent, isolated-word recognition system on a data set extracted from Switchboard telephone conversations and on a set of German sentences. We tried three different methods for creating word models (models that estimate the probability of the speech acoustics given the words). The difficulties encountered with the each model lead to refinements incorporated in the subsequent model.

We currently have word recognition rates of around 40% on the training set, and we can expect recognition performance to be worse on the test set. It is difficult to compare the results we have to other results because the training and testing sets are so different from anything that has been used before, but these results are very probably worse than state-of-the-art systems. As mentioned above, state-of-the-art, speaker-independent, isolated-word recognizers working on the Switchboard data set (from which our data set was extracted) show recognition rates around 60%-65%. However, our task is much more limited than current speech recognition work, being speaker-dependent, isolated-word recognition instead of speaker-independent, continuous-speech recognition. It might be tempting to compare our results to other speaker-dependent, isolated-word recognition tests, but our data came from the Switchboard data set, which is one of the most difficult data sets yet used for recognition evaluation.

There are several factors that contribute to our low recognition performance. The biggest disadvantage our system has compared to state-of-the-art systems is the lack of 30 years of parameter tweaking. We have made a great number of arbitrary decisions without the time to evaluate the consequences of the decisions. Just performing further parametric tests (i.e. further adjusting the dimensions and cut-off frequency of the continuity map to optimize performance) would, in all likelihood, greatly improve recognition performance. Six other factors contributing to low recognition performance are listed next. First, our training set is much smaller than the speaker-independent continuous-speech recognition training sets commonly used today (we use about 3 minutes of speech as opposed to, say, 65 hours on the complete Switchboard training set). Second, since we are doing isolated-word recognition, we are unable to take advantage of a language model. Third, the model that estimates the probability of sequences of phonemes given a word is more simplistic than in state-of-the-art recognition systems. Fourth, our current dictionary contains only canonical pronunciations of words as opposed to pronunciations that commonly occur in casual speech. This problem is particularly severe since the word extraction process sometimes deletes phonemes or adds phonemes to the beginning or the end of the word. Fifth, we are not currently using any methods for combining the outputs of different recognizers. Sixth, we did not use cepstral mean subtraction or variance normalization.

In the course of doing this work, we also performed preliminary tests on compressing speech from sentences spoken by a German speaker -- our best estimates suggest that we may be able to achieve around a 25% reduction in the number of bits needed to transmit vector quantization codes (which is already highly compressed speech). This could result in significant cost saving in military and civilian communications.

In summary, we tested more word models than we proposed and came up with a better model than we expected. Furthermore, we found evidence that MALCOM may be useful for speech compression as well. While recognition performance in the first year was not good, we have every reason to believe that the system can be improved just by using components currently used in other speech recognition systems.

2. Introduction

In this section, we present the motivation for performing the work carried out over the last year and also give an overview of the various sections of the report.

2.1 Motivation

Hidden Markov models (HMM's) are among the most popular tools for performing computer speech recognition (see Huang, Ariki & Jack, 1990). One of the primary reasons that HMM's typically outperform other speech recognition techniques is that the parameters used for recognition are determined by the data, not by preconceived notions of what the parameters should be. This lets HMM's deal with intra- and inter-speaker variability despite our limited knowledge of how speech signals vary and despite our often limited ability to correctly formulate rules describing variability and invariance in speech. In fact, it is often the case that when HMM parameter values are constrained using our (possibly inaccurate) knowledge of speech, recognition performance decreases.

Nonetheless, many of the assumptions underlying HMM's are known to be inaccurate, and improving on these inaccurate assumptions within the HMM framework can be computationally expensive (Lee, 1989, p. 142). We argue that by using probabilistic models that more accurately embody the process of speech production, we can create models that have all the advantages of HMM's, but that should more accurately capture the statistical properties of real speech samples -- leading to more accurate speech recognition.

As reviewed elsewhere (McGowan & Faber, 1996; Rose, Schroeter & Sondhi, 1996) there have been several attempts to take advantage of articulator information to improve speech recognition. Some researchers have obtained improvements in speech recognition performance by building knowledge about articulation into HMM's (Deng & Sun, 1994; Erler & Deng, 1992), or by learning the mapping between acoustics and articulation using concurrent measurements of speech acoustics and human speech articulator positions (Papcun et al., 1992; Zlokarnik, 1995). Others have worked toward incorporating articulator information by using forward models (articulatory speech synthesizers) to study the relationship between speech acoustics and articulation (McGowan & Lee, 1996; Schroeter & Sondhi, 1994).

The model we will discuss, Maximum Likelihood Continuity Mapping (MALCOM), differs from these previous attempts in that it learns the mapping from speech acoustics to articulator positions from acoustics alone -- articulator position measurements are not even used during training (Hogden, 1996; Nix, 1998). The mapping learned by MALCOM is embodied in a *continuity map*, which is a continuous, multidimensional space over which we position probability density functions -- where the probability density functions give the probability of a position in the space given an acoustic signal. Unlike Linear Predictive Coding (Markel & Gray, 1976; Wakita & Gray, 1975), which attempts to make the problem of recovering vocal tract shapes from speech acoustics tractable by using problematic simplifications (Sondhi, 1979, discusses several problems with the LPC approach), the assumptions underlying MALCOM are well-founded. In fact, the main (and surprisingly powerful) assumption used by MALCOM is that articulator motions produced by muscle contractions have little energy above 15 Hz, which is easily verified simply by calculating spectra of articulator paths (Muller & McLeod, 1982; Nelson, 1977).

The fact that MALCOM derives so much about the relationship between acoustics and articulation from so few assumptions should be a big advantage over current systems.

Consider that as we build more assumptions about articulator motions into existing models, we have a greater chance of incorporating invalid constraints and potentially decreasing recognition performance. For example, some of the rules used in existing articulation - based systems "are rather simplistic and contain several unrealistic aspects" (Deng & Sun, 1994, p. 2717). Furthermore, systems that require that information about articulation be learned from human data suffer from the fact that collecting concurrent measurements of speech acoustics and articulator positions in enough quantity to train a large vocabulary speech recognition algorithm is currently impractical. Even systems that avoid the need for human data by using articulatory speech synthesizers may be adversely affected by inaccurate assumptions -- the mapping between speech acoustics and speech articulation for articulatory speech synthesizers is strongly dependent on assumptions underlying the synthesizers and appears to differ in important ways from the mapping observed for human speech production (Hogden et al., 1996).

Given this information, it is natural to apply MALCOM to speech recognition. However, replacing HMMs for speech recognition is an ambitious goal -- HMMs have been used for recognition for about 30 years. Not only have HMMs been extensively engineered to optimize performance, but a considerable amount of work has been done to develop additional components for speech recognition (e.g. language models) that combine conveniently with HMMs. Thus, we proposed to study MALCOM for speaker-independent, isolated-word recognition, which is an easier task than the speaker-independent, continuous-speech recognition problem at the forefront of technology.

Limiting the domain of the recognition problem allowed us to focus on one aspect of speech recognition: the word model. A word model is a stochastic model used to estimate the probability of an acoustic segment given a word. The probabilities estimated for the word models can be used with additional information to find the most probable word given the speech segment, and thereby perform word recognition. So while our goal was nominally to create a MALCOM-based speech recognition system, the most important aspect of this research is not creating the recognition system per se, with all the duplication of other group's work that would be entailed, but rather to study MALCOM as an alternative word-level acoustic model.

2.2 Overview of the research

Before the current project, ad-hoc constraints had been placed on the continuity maps found by MALCOM. These constraints were necessary because without them, MALCOM produced continuity maps having fewer dimensions than desired. Since the causes of these *degenerate solutions* were not understood, it was impossible to know whether the ad-hoc constraints were sufficient, or whether further problems would occur in subsequent research. One of the first successes of the project was to refine the mathematical analysis of the MALCOM algorithm, allowing the forces that cause degenerate solutions to be understood, and showing that the ad-hoc solutions were sufficient to avoid future degeneracy problems. Thanks to the funding supplied by this project, the description of MALCOM given in Section 3 starting on page 9 is considerably more detailed than previous descriptions, and contains an explanation of the degeneracy problem.

Our sponsor asked us to use a very difficult data set (derived from the Switchboard data) to train and test our system. The methods for extracting a speaker-dependent, isolated word data set from the speaker-independent, continuous-speech Switchboard data are described in Section 4.1 starting on page 21. It took several months to actually obtain the Switchboard data, get familiar with it, and extract a usable portion. While waiting for the Switchboard data to be extracted, we studied a German speech data set. The German data set is described in Section 4.2 which starts on page 26.

As mentioned previously, the main goal of this project was to develop and test possible forms of "word models". At the beginning of this project, MALCOM had only been developed to the point where it could be used to estimate the probability of a sequence of vector quantization codes given a model of speech from a particular speaker. There was no clear idea of what kind of word model should be used. Word models have changed significantly during the 30 years that HMM recognition schemes have been used, so it should not be surprising that, over the course of only one year, it was necessary to discard several forms of MALCOM word model before getting a reasonably good one. In fact, we discarded three word models before arriving at our final word model. Brief descriptions of the discarded word models and their problems are given in Sections 5, 6, and 7. A description of the final word model is given in Section 8 which starts on page 32.

An important aspect of our final word model, the way it can be incorporated into the lattice structures commonly used today, is first described in the discussion of the next-to-last word model: Section 7.1 which starts on page 31. It came as a surprise to us that we could find a MALCOM-based word model that can easily be used in place of state-of-the-art word models. In fact, there was nothing in the original proposal or the revised proposal that indicated this was a goal.

While we were pleased that our word model could be used with state-of-the-art lattice structures, this development lead to significant changes in the recognition system. In fact, our current recognition system looks like a much simplified version of state-of-the-art recognition systems, except with a MALCOM-based word model. The description of the final recognition system (given in Section 9, starting on page 36) will make it clear that the lattices we used in the word models were simplistic, the dictionary was incomplete, and several aspects of the system could easily be modified to improve recognition performance. The reason that the recognition system is simple is that it was impossible, in the course of one year, to duplicate a state-of-the-art recognition system and also invent a new word model, particularly when we did not expect to need most of the components used in state-of-the-art systems.

The final word model uses a variant of MALCOM called Multiple-Observable MALCOM (MO-MALCOM) invented during the course of this project. Like MALCOM, MO-MALCOM creates a mapping from a continuous, underlying, abstract space (called a continuity map) to acoustics. Also like MALCOM, positions in the continuity map can be shown to be related to articulator positions. However, MO-MALCOM continuity maps have a very important difference: they are designed so that positions in the continuity map maximally discriminate phonemes. In fact, this difference is probably the single biggest success to come out of this project. If the reader chooses to read only one section of this document, we hope that it will be Section 8.3 starting on page 34, showing that MO-MALCOM's ability to discriminate phonemes appears to be on par or better than the ability of articulator positions to discriminate phonemes. This finding is important because recent work by Roweis at Caltech (Roweis, 1998) shows that measured articulator positions can be used in conjunction with acoustics to get near perfect recognition on a speaker-dependent task. It is also MO-MALCOM's ability to discriminate phonemes that argues that our current speech recognition performance is not limited by the MALCOM-based word model, but is instead limited by the unsophisticated nature of the non-MALCOM components of our recognition algorithm.

While not specifically part of the project, in the course of studying speech recognition, we obtained results that strongly suggest that MALCOM is applicable to speech compression. These results, which we believe will be of interest, are described in Section 11 starting on page 38.

3. MALCOM

3.1 Basic Definitions

Maximum Likelihood Continuity Mapping (MALCOM), learns the mapping from speech acoustics to pseudo-articulator positions from acoustics alone. Articulator position measurements are not used even during training. A mapping is found between pseudo-articulator positions and probability density functions (PDFs) over an abstract space called a *continuity map*. The word "mapping", as used herein, is meant in the mathematical sense. Informally, a mapping is a rule that assigns to each object in one set a unique object in another set. Thus, a unique probability density function over positions in a continuity map is assigned to each sound type. Any such mapping (i.e., from speech sounds to probability density function over a continuity map) will be referred to as a probabilistic mapping. MALCOM learns, among other things, a particular probabilistic mapping, referred to as a maximum likelihood continuity mapping. Note the difference between a "mapping" and a "map" as used herein: "map" refers to an abstract space that may or may not include probability density functions, but a "mapping" defines a relationship between two sets. As a final note on usage, "mapping" may be used as either a noun or a verb. The usage context will make the distinction evident.

The application of MALCOM to speech processing will be described in part through comparisons between MALCOM and prior art HMM speech recognition systems. Thus, a brief description of HMM techniques will be provided first, leading into a discussion of the basic MALCOM algorithm.

3.2 Brief introduction to hidden Markov models

In a straightforward implementation of the prior art HMM approach, models are made of each word in the vocabulary. The word models are constructed such that the probability that any acoustic speech sample would be produced given a particular word model can be determined. The word model most likely to have created a speech sample is taken to be the model of the word that was actually spoken. For example, suppose some new speech sample, **Y**, is produced. If w_i is the model for word i, and w_i maximizes the probability of **Y** given w_i, then a HMM speech recognition algorithm would take word i to be the word that was spoken. In other variants of HMM speech recognition, models are made of phonemes, syllables, or other subword units.

Figure 1 shows a classic 5 state HMM. Each of the circles in Figure 1 represents an HMM state. At any time, the HMM has one active state and a sound is assumed to be emitted when the state becomes active. The probability of sound y being emitted by state s_i is determined by some parameterized distribution associated with state s_i (e.g. a multivariate Gaussian parameterized by a mean and a covariance matrix). The connections between the states represent the possible interstate transitions. For example, in the left-to-right model shown in Figure 1, if the model is in state s_2 at time t, then the probability of being in state s_4 at time $t+1$ is a_{24}.

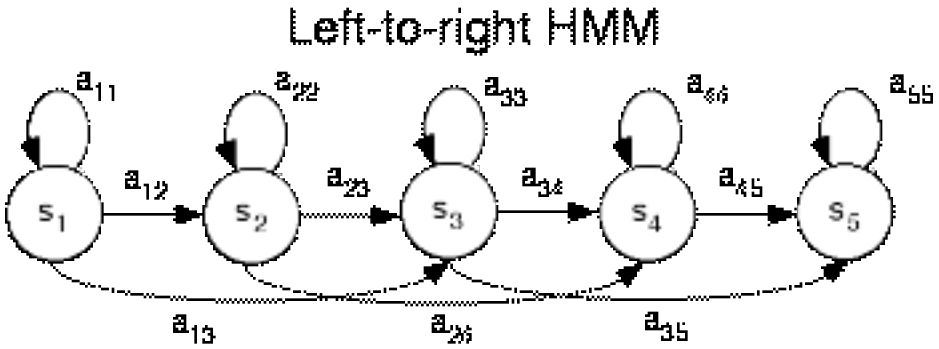


Figure 1

HMM's are trained using a labeled speech data base. For example, the data set may contain several samples of speakers producing the word "president". Using this data, the parameters of the "president" word model (the transition probabilities and the state output probabilities) are adjusted to maximize the probability that the "president" word model will output the known speech samples. Similarly, the parameters of the other word models are also adjusted to maximize the probability of the appropriate speech samples given the models. As the word models more closely match the distributions of actual speech samples (i.e. the probability of the data given the word models increases), the recognition performance will improve, which is why the models are trained in the first place.

3.3 The basic MALCOM algorithm

MALCOM provides better estimates of the distributions of speech data by basing the acoustic models on the actual processes underlying speech production. Consider that speech sounds are produced by slowly moving articulators. If the relationship between articulator positions and speech acoustics is known, information about the articulator positions preceding time *t* can be used to accurately predict the articulator positions at time *t*, and therefore better predict the acoustic signal at time *t*.

MALCOM uses a model of sequence generation (acoustic sequences or otherwise) in which symbols are produced as a point moves through an abstract space called a continuity map (CM). Figure 2 shows a hypothetical continuity map that will be used to explain MALCOM. The CM in Figure 2 is characteristic of a CM used to determine the probability of sequences composed of symbols in the set {1, 2, 3, 4}, such as the sequence {1, 4, 4, 3, 2}. In Figure 2, the concentric ellipses around the number "2" are used to represent equiprobability curves of a probability density function (PDF). The PDF gives the probability of a CM position given the symbol "2". Similarly, the ellipses surrounding the numbers 1, 3, and 4, represent equiprobability curves of PDFs giving the probability of CM positions given these other symbols. For ease of exposition, assume that the PDFs are Gaussians, even though Gaussians are not required to use the MALCOM algorithm.

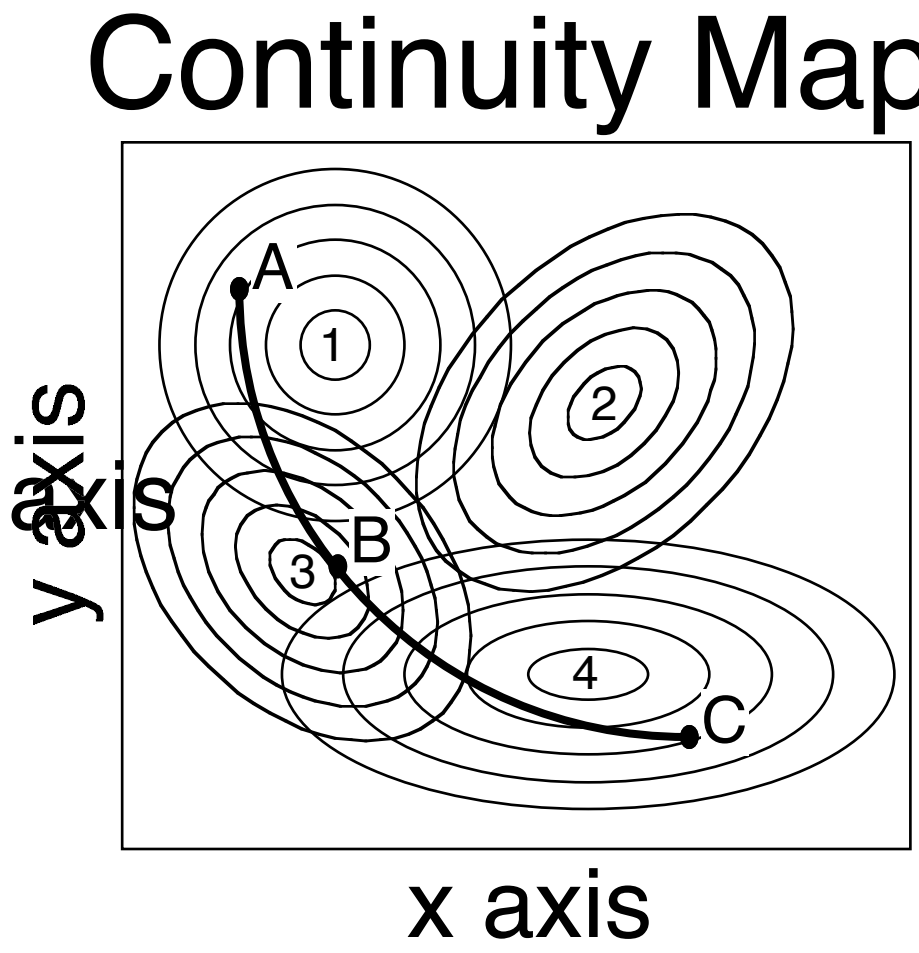


Figure 2

Note that the CM axes are meaningless in this case, and so are labeled simply "axis x" and "axis y". All that matters is the relative positions of the objects in the CM. In fact, it will later be shown that any rotation, reflection, translation, or scaling of the positions of the objects in the CM will be an equally good CM.

To find the probability of a sequence of symbols using MALCOM, the sequence of positions in the CM that maximizes the probability of the symbol sequence is found. If all paths through the CM were equally probable, then all sequences which differed only in the order of the symbols would be equally probable. To see why, note that the path which maximizes the probability of the sequence {1, 3, 4} goes (approximately) from the mode of the PDF for symbol "1", to the mode of the PDF for symbol "3', to the mode of the PDF for symbol "4". The path through the CM which maximizes the probability of the sequence {3, 1, 4} goes through the same points just in a different order. From this fact, it is possible to show that the probability of sequence {1, 3, 4} given the first path through the CM will be exactly equal to the probability of sequence {3, 1, 4} given the second path through.

Since it is important to be able to represent the fact that symbol sequences differing in order may have different probabilities, MALCOM constrains the possible paths through the CM. As the smooth curve connecting the points "A", "B", and "C" suggests, MALCOM as currently embodied requires that paths through the CM are smooth, a physical constraint of articulatory motion. This smoothness constraint could easily be replaced by other

constraints for other applications of MALCOM (e.g. that the probability of a path goes down as the frequencies increase, or that the paths must all lie on a circle, etc.).

In order to determine the probability of a sequence, MALCOM is used to adjust the parameters of the PDFs associated with the symbols in a manner that maximizes the probability of all the data sequences in a known training set. However, since the algorithm for adjusting the PDF parameters uses the technique for finding the path that maximizes the probability of the data, the following description will first discuss how the best path is found given a probabilistic mapping, and then discuss how to make a maximum likelihood continuity mapping.

While we will discuss data sets consisting of vector quantization (VQ) codes (Gray, 1984 describes vector quantization) derived from speech acoustics, there is nothing in the underlying MALCOM theory that requires the symbols to be in any way related to speech.

3.3.1 Finding maximum likelihood paths

As discussed above, MALCOM finds a mapping between speech acoustics and positions in a continuity map. Since continuity map positions are related to articulator positions, we often refer to continuity map positions as pseudo-articulator positions. This section and the flow charts included at the end of this document show how to determine pseudo-articulatory paths corresponding to sound sequences.

The probability of a sequence of speech sounds given a pseudo-articulatory path will be derived by first finding the probability of a single speech sound given a single pseudo-articulator position, and then by combining probabilities over all the speech sounds in a sequence. Next, a technique for finding the pseudo-articulator path that maximizes the conditional probability of a sequence of speech sounds will be described. Finally, the problem is constrained to find the smooth pseudo-articulator path (as opposed to any arbitrary pseudo-articulator path) that maximizes the conditional probability of the data.

The following definitions are used. Let:

$c(t)$ = the VQ code assigned to the t^{th} window of speech;

$\mathbf{c} = [c(1), c(2), \ldots c(n)]$ = a sequence of VQ codes used to describe a speech sample, where n is the number of VQ codes used;

$x_i(t)$ = the position of pseudo-articulator i at time t;

$\mathbf{x}(t) = [x_1(t), x_2(t), \ldots x_d(t)]$ = a vector of the positions of all the pseudo-articulators at time t where d is the number of dimensions in the CM; and

$\mathbf{X} = [\mathbf{x}(1), \mathbf{x}(2), \ldots \mathbf{x}(n)]$ = a sequence of pseudo-articulator configurations.

Further definitions are needed to specify the mapping between VQ codes and PDFs over pseudo-articulator positions. Let:

$P(c_i)$ = the probability of observing code c_i given no information about context;

$p(\mathbf{x}|c_i,\theta)$ = the height of the probability density function giving the probability density that pseudo-articulator position $\mathbf{x}$ was used to produce VQ code c_i ;

θ = a set of model parameters that define the shape of the PDF, e.g., θ could include the mean and covariance matrix of a Gaussian probability density function used to model the distribution of $\mathbf{x}$ given c.

Note that an uppercase "P" is used to indicate a probability, whereas a lowercase "p" is used to indicate the height of a probability density function -- a convention used throughout this report.

Many different distributions could be used for $p(\mathbf{x}|c_i,\Theta)$. For example, computer simulations have been used to argue that many different articulator positions produce the same acoustic signal. Although the limited research on human speech production data argues that articulator positions can be recovered from acoustics much more accurately than computer simulations suggest, if there are multimodal distributions of articulator positions that can be used to produce identical acoustic signals, then it may be necessary to specify $p(\mathbf{x}|c_i,\Theta)$ as a mixture of Gaussians. In the current work, only simple Gaussians were explored, leaving open the possibility of improving recognition performance in the future.

With these definitions, the probability of observing code c_j, given that the pseudo-articulator position vector is $\mathbf{x}$ with model parameters Θ, is determined using Bayes' Law as:

Equation 1
$$P(c_j|\mathbf{x},\Theta) = \frac{p(c_j,\mathbf{x}|\Theta)}{p(\mathbf{x}|\Theta)} = \frac{p(c_j,\mathbf{x}|\Theta)}{\sum_i p(c_i,\mathbf{x}|\Theta)} = \frac{p(\mathbf{x}|c_j,\Theta)P(c_j)}{\sum_i p(\mathbf{x}|c_i,\Theta)P(c_i)}$$

The probability of the code sequence can be determined by assuming conditional independence, i.e.,

Equation 2
$$P[\mathbf{c}|\mathbf{X},\Theta] = \prod_{t=0}^{n} P[c(t)|\mathbf{x}(t),\Theta]$$

Conditional independence implies that the probability of producing a given sound, or VQ code, is wholly dependent on the current tongue position without any regard to the previous tongue position. Note that this is a realistic assumption for speech: the sound produced using a particular vocal tract shape can be determined just from the vocal tract shape. Further note that the probability of observing a code is not assumed to be independent of the preceding and subsequent codes; it is only assumed to be conditionally independent. So if $\mathbf{x}(t)$ is dependent on $\mathbf{x}(t')$ then $c(t)$ is dependent on $c(t')$. By using an appropriately constrained model of possible pseudo-articulator paths the sequences of codes can be tightly constrained in a biologically plausible manner.

The goal is to find the sequence of pseudo-articulator positions $\mathbf{X}$ that maximizes the conditional probability, $P(\mathbf{c}|\mathbf{X},\Theta)$, of a sequence of VQ codes, $\mathbf{c}$. A succinct notation for writing "Let $\hat{\mathbf{X}}$ be the $\mathbf{X}$ that maximizes $P(\mathbf{c}|\mathbf{X},\Theta)$ is:

Equation 3
$$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} P(\mathbf{c} \mid \mathbf{X},\Theta)$$

Function maximization is such a useful process that many standard maximization algorithms already exist. While many of the standard algorithms could be used, the more efficient algorithms require calculating the gradient of the function to be maximized. Thus, the gradient with respect to $\mathbf{X}$ is derived here.

To simplify the problem of finding $\hat{\mathbf{X}}$, we note that the $\mathbf{X}$ that maximizes $P(\mathbf{c}|\mathbf{X},\Theta)$ also maximizes $LogP[\mathbf{c} \mid \mathbf{X},\Theta]$. Thus, $LogP[\mathbf{c} \mid \mathbf{X},\Theta]$ is maximized using the gradient of $LogP[\mathbf{c} \mid \mathbf{X},\Theta]$, which is denoted $\nabla LogP[\mathbf{c} \mid \mathbf{X},\Theta]$, to get the maximum likelihood estimate of the pseudo-articulator path that produced $\mathbf{c}$.

This gradient is found by first taking the logarithm of Equation 2:

Equation 4 $$LogP[\mathbf{c}|\mathbf{X},\theta] = \sum_t LogP[c(t)|\mathbf{x}(t),\theta]$$

Next, substitute Equation 1 into Equation 4 and separate the terms in the logarithm to get:

Equation 5

$$LogP[\mathbf{c}|\mathbf{X},\theta] = \sum_t \left(Logp[\mathbf{x}(t)|c(t),\theta] + LogP[c(t)] - Log\sum_i p[\mathbf{x}(t)|c_i,\theta]P[c_i] \right)$$

The gradient of equation 5 is:

Equation 6 $$\nabla LogP[\mathbf{c}|\mathbf{X},\theta] = \frac{\nabla p[\mathbf{x}(t_0)|c(t_0),\theta]}{p[\mathbf{x}(t_0)|c(t_0),\theta]} - \frac{\sum_i P[c_i]\nabla p[\mathbf{x}(t_0)|c_i,\theta]}{\sum_i p[\mathbf{x}(t_0)|c_i,\theta]P[c_i]}$$

The preceding analysis ignores constraints on the possible pseudo-articulator paths. To incorporate biologically plausible constraints on pseudo-articulator motion, only those pseudo-articulator paths are allowed that have all their energy in Fourier components below some cut-off frequency (say 15 Hz, since actual articulator paths have very little energy above 15 Hz). Realizing that a discrete Fourier transform can be considered a rotation to an new set of orthogonal basis vectors, the constraint that the pseudo-articulator path have all of its energy below the cut-off frequency is equivalent to requiring that the path lie on a hyperplane composed of the axes defined by low frequency sine and cosine waves.

From vector calculus, when ∇LogP($\mathbf{c}|\mathbf{X},\theta$) is perpendicular to the constraining hyperplane, i.e., has no components below the cut-off frequency, so that LogP($\mathbf{c}|\mathbf{X},\theta$) can not increase without $\mathbf{X}$ traveling off the hyperplane, then a constrained local minimum has been reached. Thus, the smooth path that maximizes the probability of the observed data is the path for which ∇LogP($\mathbf{c}|\mathbf{X},\theta$) has no components with energy below the cut-off frequency. This suggests the following process for finding the smooth path that maximizes the probability of the data, as shown in Flow Chart 1:

1) read 12 a data sequence;

2) initialize the maximization algorithm by choosing 14 a pseudo-articulator path and low-pass filtering 16 the initial path selected.

3) use standard maximization techniques 18 (e.g., conjugate gradient descent in conjunction with subroutine 22 that calculates 24 the gradient and low-pass filters 26 the gradient) to find the smooth path that maximizes the probability of the data given the path;

4) let the maximization algorithm converge 18 to a solution;

5) store 28 the most likely smooth path obtained for the data sequences:

6) repeat 32 steps 12-28 until all of the paths have been read.

3.3.1.1 A note on initialization

While, theoretically, any smooth path could be used to initialize 14 the maximization algorithm, a solution can be found more quickly if a good initial smooth path can be found.

In fact, for many classes of distributions of $p(\mathbf{x}|c_i,\theta)$ a good initial path can be found. To explain how to find a good initial path, assume the code sequence [5, 2, ...] is observed. A path is created by using the mean pseudo-articulator position associated with code 5 as the first point in the path, then the mean pseudo-articulator position associated with code 2 as the second point in the path, etc. Finally, the path is low-pass filtered to ensure that it is smooth.

3.3.1.2 A note on low-pass filtering

Since it may not be clear what is meant by low-pass filtering a multidimensional path, the goal of this section is to describe this operation in more detail. A way to filter paths is described which will give the same low pass-filtered results regardless of rotations, reflections, or translations of the paths. Consider that, in a path through a d-dimensional space, there are d measurements at each time. It can be shown that low-pass filtering the time series composed of $[x_1(1), x_1(2), \ldots x_1(n)]$ and then low-pass filtering the path composed of $[x_2(1), x_2(2), \ldots x_2(n)]$, etc., until the path has been low-pass filtered on each dimension, will force the path to be low-pass filtered regardless of any rotation, reflection, or scaling of the CM. This result follows because any linear combination of signals having no energy in Fourier components above f_c will have no energy in Fourier components above f_c, and rotation, reflection, and scaling are all linear operations.

However, in the hypothetical continuity map shown in Figure 2, the x-axis components of the path [A,B,C] increase in value from time 1 to time 3, but the CM could easily be rotated and translated so that the x-axis component of the path at times 1 and 3 are 0 and the x-axis component of the path at time 2 is some positive value. This fact affects how the low-pass filtering is performed, because discrete-time filtering theory assumes that the paths are periodic -- after the last element of the time series, the series is assumed to restart with the first element of the time series. Thus, by performing simple rotations and translations of the CM, time series are obtained that have large discontinuities or are relatively smooth.

To avoid problems that would arise from the discontinuities, the trend or bias of the time series is removed before smoothing the paths and then added back after the filtering has been performed, i.e., the line connecting the first and last points in the path should be subtracted from the path before filtering and added back after filtering. The trend should also be removed before filtering the gradient and then added back after filtering the gradient.

These steps are depicted in Flow Chart 5 and comprise the low-pass filter process:

1) select 40 a first dimension (set d=1);

2) project 42 a path/gradient onto dimension d to find the length of the projection of the path position onto dimension d at each time to get a time series composed of a scalar value of each time;

3) remove 43 the trend of the projected path/gradient, i.e., subtract from the time series values the height of a straight line joining the first and last points in the time series;

4) low-pass filter 46 the projection, less the trend, of the path/gradient onto dimension d;

5) add 47 the trend back to the path/gradient

6) determine 48 if dimensions remain;

7) if so, increment 52 to the next dimension and repeat; if not, complete the process.

3.3.2 Finding a mapping between pseudo-articulation and acoustics

In the preceding section, it is assumed that $P(c)$ and $p(\mathbf{x}|c,\Theta)$ are known. In this section it is shown that these values can be determined using only acoustic data. This is an important aspect of MALCOM, because $p(\mathbf{x}|c,\Theta)$ is a probabilistic mapping from speech sounds to pseudo-articulator positions. The techniques in this section allow a mapping between pseudo-articulator positions and acoustics to be obtained, without inputting possibly faulty knowledge of phonetics into a model, without collecting measurements of articulator positions, and without using potentially inaccurate articulatory synthesizers to learn the mapping from acoustics to articulator positions.

The process of finding the mapping between speech sounds and PDFs over continuity map positions is presented in the flow charts. Flow Chart 2 shows the steps needed to learn the mapping:

1) given a collection of quantized speech signals and some initial estimate 62 of the mapping, use the procedures described herein to find the smooth paths 64 (see Flow Chart 1) (one path per sequence) that maximize the conditional probability of the observed data, i.e., for each sequence find:

$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} P(\mathbf{c} \mid \mathbf{X},\Theta)$ where $\hat{\mathbf{X}}$ is constrained to be smooth;

2) given the smooth paths that maximize the probability of the data sequences, find 66 the PDF parameters, Θ (Flow Chart 4) that maximize (or at least increase) the conditional probability of the data set, i.e., find:

$$\hat{\Theta} = \arg\max_{\Theta} \prod_{\mathbf{c}} P\left(\mathbf{c}|\hat{\mathbf{X}},\Theta\right) \text{ and } \hat{P}(c_i) = \arg\max_{\hat{P}(c_i)} \prod_{\mathbf{c}} P\left(\mathbf{c}|\hat{\mathbf{X}},\Theta\right)$$

where the products are taken over all data sequences. As discussed below, the $P(c_i)$ values are calculated from the number of each code in the data set. An implication of this is that the $P(c_i)$ values that maximize the conditional probability of the data do not change, and so can be calculated once, at the beginning of the algorithm, as part of the initialization 92-96.

3) Impose 67 on Θ any PDF-dependent additional constraints needed for the particular probability density function used;

4) determine 68 the difference between the values from step 67 and current values;

5) if the difference is not below a threshold difference, replace 72 the previous Θ with the new Θ and repeat steps 64-67 iteratively until a local (possibly global) probability maximum is reached;

6) the Θ that is a local maximum is then stored 74 and the process is completed 76..

Calculation of Θ can be accomplished using standard maximization algorithms. Since maximization algorithms that use gradient information are typically faster than algorithms that do not use the gradient, an expression for ∇LogP($\mathbf{c}|\mathbf{X},\Theta$) with respect to Θ is derived to aid in maximizing the probability of the data:

$$\nabla LogP[\mathbf{c}|\mathbf{X},\theta]$$

$$= \nabla \sum_t \left(LogP[\mathbf{x}(t)|c(t),\theta] + LogP[c(t)] \right) - \nabla Log \sum_i P[\mathbf{x}(t)|c_i] P[c_i]$$

$$= \sum_t \left(\frac{\nabla P[\mathbf{x}(t)|c(t),\theta]}{P[\mathbf{x}(t)|c(t),\theta]} + \frac{\nabla P[c(t)]}{P[c(t)]} - \frac{\sum_i \nabla \{P[\mathbf{x}(t)|c_i,\theta] P[c_i]\}}{\sum_i P[\mathbf{x}(t)|c_i,\theta] P[c_i]} \right)$$

concluding with:

Equation 7

$$\nabla LogP[\mathbf{c}|\mathbf{X},\theta] = \sum_t \left(\frac{\nabla P[\mathbf{x}(t)|c(t),\theta]}{P[\mathbf{x}(t)|c(t),\theta]} - \frac{\sum_i \nabla \{P[\mathbf{x}(t)|c_i,\theta] P[c_i]\}}{\sum_i P[\mathbf{x}(t)|c_i,\theta] P[c_i]} \right)$$

Flow Chart 4 illustrates a process using standard techniques 82 (e.g., conjugate gradient) to find the parameters (e.g., means and covariance matrices) of the probability density functions associated with each symbol in the data set that maximized the probability of the data. Subroutine 84 calculates the gradient of the probability of all the data sequences with respect to each probability density function parameter using Equation 7.

The *P(c)* values that maximize the probability of the VQ code sequences can be found analytically. To derive the *P(c)* values, start with the expression for ∇LogP($\mathbf{c}|\mathbf{X},\theta$) with respect to $P(c_k)$:

$$\nabla LogP[\mathbf{c}|\mathbf{X},\theta] = \sum_{t \ni c(t)=c_k} \frac{1}{P[c_k]} - \sum_{t=0}^{n} \left(\frac{p[\mathbf{x}(t)|c_k,\theta]}{\sum_i p[\mathbf{x}(t)|c_i,\theta] P[c_i]} \right)$$

$$= \frac{n_k}{P[c_k]} - \sum_{t=0}^{n} \left(\frac{p[\mathbf{x}(t)|c_k,\theta]}{\sum_i p[\mathbf{x}(t)|c_i,\theta] P[c_i]} \right)$$

Equation 8

$$= \frac{n_k}{P[c_k]} - \frac{1}{P(c_k)} \sum_t P[c_k|\mathbf{x}(t),\theta]$$

$$= \frac{n_k}{P[c_k]} - \frac{1}{P(c_k)} \sum_{\mathbf{x}} p(\mathbf{x}) P[c_k|\mathbf{x},\theta]$$

$$= \frac{n_k}{P[c_k]} - 1$$

where n_k is the number of times c_k is observed in the speech sample. Since the sum of the *P(c)* values must be 1, finding the *P(c)* values that maximize the conditional probability of the data is a constrained optimization problem in which the *P(c)* values are found by using a Lagrange multiplier, λ, and solving:

Equation 9 $\frac{n_k}{P(c_k)} \cdot 1 = \cdot \sum_i P(c_i) = \cdot$

From Equation 9, it can be seen that setting

Equation 10 $P(c_k) = n_k/n$

will maximize the conditional probability of the data.

Initial parameters are established as shown in Flow Chart 3. For each symbol k in the data set, the number of occurrences, n_k, is found 92 for that symbol. Then, n=sum over k of n_k is calculated 94. The probability $P(c_k)=n_k/n$ of each symbol is set 96 (Equation 10) and an initial set of parameters (e.g., means and covariance matrices) is chosen 98 for the PDF's associated with each symbol in the data set. This establishes the initial map 62 for use in the process shown in Flow Chart 2.

Thus, using only speech acoustics, it is possible to infer a probabilistic mapping between acoustics and pseudo-articulator positions. Furthermore, given speech acoustics and said probabilistic mapping (or a probabilistic mapping created by a different method such as a mapping that is made using measured articulator positions), it is possible to find the pseudo-articulator trajectories most likely to have created the acoustics.

3.3.2.1 A note on PDF-dependent constraints

For most forms of the $p(\mathbf{x}|c,\cdot)$ distributions, there are many different $\cdot$ values that will give equally high values of $P(\mathbf{c}|\mathbf{X},\cdot)$. Some of these solutions are degenerate and should be avoided. Examples of simple constraints that can be applied when using Gaussian distributions are discussed in the example derivation given in Section 3.3.3, starting on page 18.

3.3.2.2 Choosing the dimensionality and cut-off frequency

Even though the probability of the data increases as the dimensionality and/or cut-off frequency of the MALCOM solution increase, it clearly is not the case that increasing the dimensionality or cut-off frequency will always give a better solution. While the choice of the dimensionality and cut-off frequency depends in part on the application, one aid to choosing these parameters is the number of bits needed to transmit the data. The number of bits needed to transmit the data is the sum of the number of bits needed to transmit the smooth paths and the number of bits needed to transmit the codes given the smooth paths. It is known from information theory that the number of bits needed to transmit the data given the smooth paths is $\cdot \sum_t LogP\left[c(t)|\mathbf{x}(t),\cdot\right]$. Notice that the number of bits needed to transmit the smooth paths increases with increasing dimensionality and cut-off frequency (since the number of samples per second increases), whereas the number of bits needed to transmit the data given the smooth paths decreases with increasing dimensionality and cut-off frequency. Thus, the number of bits needed to transmit the data better captures the trade-off between parsimony and accuracy.

3.3.3 Example

The above derivation permits many different forms of the $p\left[\mathbf{x}(t)|c(t),\cdot\right]$ distributions to be used. In this section the gradient equations are derived for the exemplary case where the distribution of articulator positions that produce sounds quantized by code c is a multivariate Gaussian characterized by the equation:

Equation 11

$$p[\mathbf{x} \mid c, \theta] = \frac{1}{(2\pi)^{\frac{d}{2}} |\Sigma(c)|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\left[\mathbf{x} - \mu(c)\right]^t \Sigma^{-1}(c)\left[\mathbf{x} - \mu(c)\right]\right]$$

where:

d is the number of dimensions in the pseudo-articulator space (i.e., the number of pseudo-articulators),

$\mu(c)$ is a vector giving the mean of all the pseudo-articulator positions used to produce sounds quantized with vector quantization code c. For example, $\mu_i(c)$, the i^{th} component of the $\mu(c)$ vector, may be correlated with the mean lower lip height used to create sounds quantized as code c,

$\Sigma(c)$ is the covariance matrix of the multivariate Gaussian distribution of pseudo-articulator positions that produce sounds quantized with code c, and

$\mathbf{x}$ is a vector describing a pseudo-articulator configuration.

As mentioned above, the $\mathbf{x}$, $\mu(c)$ and $\Sigma(c)$ values that maximize the conditional probability of the data are not unique. For example, suppose $\mathbf{x}$, $\mu(c)$, and $\Sigma(c)$ maximize the conditional probability of the data. Let $\mathbf{R}$ be an arbitrary matrix and let $\mathbf{y}$ be an arbitrary vector. Also let $\mathbf{x}' = \mathbf{R}\mathbf{x}+\mathbf{y}$, $\mu'(c) = \mathbf{R}\mu(c)+\mathbf{y}$, and $\Sigma'(c) = \mathbf{R}\Sigma(c)\mathbf{R}^t$. With these definitions, the probability of $\mathbf{x}'$ given a code and the model is

Equation 12

$$P[\mathbf{x}' \mid c, \theta'] = \frac{1}{|\mathbf{R}|} P[\mathbf{x} \mid c, \theta]$$

Notice that the probability is only changed by a scaling factor and goes to infinity as the determinant of $\mathbf{R}$ goes to 0. Furthermore, if Equation 12 is substituted into Equation 1, it can be seen that the conditional probability of the VQ codes is the same for $\mathbf{x}'$, $\mu'(c)$ and $\Sigma'(c)$ as it was for $\mathbf{x}$, $\mu(c)$ and $\Sigma(c)$. Thus, an infinite number of solutions will all be equally good if there are no additional constraints placed on the solutions. Among the solutions that are equally good are rotations, reflections, translations, and scalings of the configuration of $\mu(c)$ values. While rotations, reflections, scalings, and translations of the solutions are inconsequential, numerical difficulties can occur if the determinant of $\mathbf{R}$ goes to zero. For this reason, it is a good idea to place additional constraints on the solution to prevent these "degenerate" solutions.

There are a variety of simple constraints that can be used to prevent degenerate solutions. In this discussion, we will treat the $\mathbf{x}$, $\mu(c)$ and $\Sigma(c)$ values as the "correct" values that correspond to quantities in the underlying production system and $\mathbf{x}'$, $\mu'(c)$ and $\Sigma'(c)$ will be taken to be the estimated values obtained by MALCOM. One way to constrain the solutions is to require $\Sigma'(c)$ to be the identity matrix for at least one value of c. This forces $\mathbf{R} = \Sigma^{-1/2}(c)$, which is sufficient to prevent the determinant of $\mathbf{R}$ from being 0 as long as $\mu(c)$ has full rank.

Alternately, since the constraint on $\Sigma'(c)$ is not guaranteed to prevent a degenerate solution, we can use a constraint on the $\mu'(c)$ values. For example, if we let $\mathbf{v}_i = \left[\mu_i(c_1) \quad \mu_i(c_2) \quad \cdots \quad \mu_i(c_m)\right]$ where i indexes the components of the $\mu'(c)$ vector and m is the number of symbols in the vocabulary (i.e., the number of distinct VQ codes), then

we can ensure that **R** has full rank by first forcing the components of each $\mathbf{v}_i$ to sum to 0 (by subtracting the means), then by using Gram-Schmidt orthogonalization to force the $\mathbf{v}_i$ to be mutually orthogonal, and finally scaling the $\mathbf{v}_i$ to all be length 1. If these steps are performed after each re-estimation of Φ, the solutions will only differ by rotations and reflections, which are irrelevant. Of course, combinations of constraints can also be used. While using combinations of constraints will overconstrain the solution, it will also decrease the number of parameters that need to be estimated and thereby potentially lead to better solutions with limited data sets.

Returning to the problem of finding the gradient equations for the Gaussian probability density function, let ∇ denote the gradient with respect to the components of **x**, so

Equation 13 $$\nabla p[\mathbf{x}|c,\Phi] = \nabla p[\mathbf{x}|c,\Phi]\Sigma^{-1}(c)[\mathbf{x} - \mu(c)],$$

which can be substituted into Equation 6 to aid in finding the path that maximizes the conditional probability of the data:

$$\nabla LogP[\mathbf{c}|\mathbf{X},\Phi] = \sum_t \Sigma^{-1}[c(t)]\{\mathbf{x}(t) - \mu[c(t)]\}$$

Equation 14

$$+\frac{\sum_i P[c_i]\, p[\mathbf{x}(t)|c_i,\Phi]\Sigma^{-1}(c_i)\{\mathbf{x}(t) - \mu(c_i)\}}{\sum_i p[\mathbf{x}(t)|c_i,\Phi]P[c_i]}$$

Similarly, the gradient with respect to $\mu(c_k)$ is:

Equation 15 $$\nabla p[\mathbf{x}|c_i,\Phi] = p[\mathbf{x}|c_i,\Phi]\Sigma^{-1}(c_i)[\mathbf{x} - \mu(c_i)]\delta_{ik} \qquad \delta_{ik} = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

Which, finally, can be substituted into Equation 7 to get:

$$\nabla LogP[\mathbf{c}|\mathbf{X},\Phi] = \sum_{t\ni c(t)=c_k} \Sigma^{-1}[c(t)]\{\mathbf{x}(t) - \mu[c(t)]\}$$

Equation 16

$$\sum_t \frac{P[c_k]p[\mathbf{x}(t)|c_k,\Phi]\Sigma^{-1}(c_k)\{\mathbf{x}(t) - \mu(c_k)\}}{\sum_i p[\mathbf{x}(t)|c_i,\Phi]P[c_i]}$$

3.3.3.1 An important simplification

There is a simplified version of MALCOM (MALCOM *1*) that is actually a very close approximation to the correct algorithm and runs much faster To be more specific, if we let C denote the number of VQ codes in the data set, then the run time for finding a path with MALCOM is $O(C^2)$ whereas the run time for MALCOM 1 is O(C). In one problem with 256 VQ codes, MALCOM 1 calculated an approximation to the most likely smooth path in about 1/125th of real time, whereas MALCOM required 1/4 real time to calculate a smooth path that differed very little from the MALCOM 1 path. This difference in speed is magnified when trying to estimate the continuity map parameters since each path must be calculated many times. This suggests that solutions obtained by MALCOM *1* should be

used in place of or to initialize the MALCOM procedure for finding the mapping between acoustics and pseudo-articulator positions.

Instead of maximizing the conditional probability of the observed data, MALCOM *1* recovers the mapping between acoustics and articulation by maximizing the probability of the smooth articulator paths. Mathematically, this amounts to ignoring the second term in Equation 14 and Equation 16. The simplified versions of Equation 14 and Equation 16 are, respectively:

Equation 17 $$\nabla LogP\left[\mathbf{c}|\mathbf{X},\theta\right] = \Sigma_{c(t)}^{-1}\left[c(t)\right]\left\{\mathbf{x}(t) - \mu\left[c(t)\right]\right\}$$

and

Equation 18 $$\nabla LogP\left[\mathbf{c}|\mathbf{X},\theta\right] = \sum_{t} \Sigma_{c(t)}^{-1}\left[c(t)\right]\left\{\mathbf{x}(t) - \mu\left[c(t)\right]\right\}$$

Notice that Equation 18 is significantly simpler to maximize than Equation 16. Equation 16 requires an iterative maximization algorithm whereas Equation 18 can be solved analytically. The analytic solution for Equation 18 sets

Equation 19 $$\mu(c_i) = \sum_{t \ni c(t)=c_i} \mathbf{x}(t) \Big/ n_i$$

P(c) is not calculated in MALCOM *1* because no information about *P(c)* can be extracted without trying to maximize the conditional probability of the data instead of the probability of the smooth paths. For this study, all the covariance matrices were set to the identity matrix.

The degeneracy problem is much worse when using MALCOM *1* than it is when using MALCOM. The problem is worse because MALCOM *1* maximizes the probability of the smooth paths, and as discussed above, these probabilities go to infinity as the determinant of **R** goes to 0. Thus, without imposing constraints, MALCOM *1* will return degenerate solutions if allowed to run indefinitely. In the following description, the means were constrained with centering, orthogonalizing, and scaling, as discussed above.

4. Data sets

4.1 Isolated Words from Switchboard

Extracting an isolated-word data set from a continuous speech data set is not straightforward. In fact, a couple months of time went into creating the data set. This not only makes this data set fairly expensive, but it also decreased the amount of progress made on the main goal of this project: exploring MALCOM-based word models.

As it exists now, the isolated-word data set contains many errors (discussed in Section 4.1.3). Given sufficient time, the data set could have been made perfect, but doing so would have been prohibitively expensive, especially considering that this data set is unlikely to be extensively studied.

4.1.1 Data Set Description

Derived from the Switchboard corpora of telephone conversations, the data set is composed of digitized acoustics and time-aligned phonetic transcriptions of isolated words extracted

from the conversations of one male (speaker 1245) and one female (speaker 1260). The acoustics are stored as linearly quantized 16 bit integers, sampled at 8 kHz. The phonetic transcriptions are stored in the standard NIST format.

Phonetic transcription are only available for some of the extracted words. Speaker 1245 was chosen because he is the male speaker in the Switchboard corpus with the most phonetically transcribed speech. Similarly, speaker 1260 is the female speaker with the most phonetically transcribed speech. Information about the 14 speakers with the most phonetically labeled speech is given in Table 1.

Speaker ID	Gender	Dialect	Time, sec
1260	F	North Midland	422
1086	F	Southern	246
1465	F	NYC	218
1245	M	Southern	180
1175	F	Western	179
1237	M	North Midland	142
1312	M	Southern	121
1087	M	Northern	112
1010	M	New England	80
1512	F	South Midland	77
1249	F	Western	77
1063	M	North Midland	67
1126	F	South Midland	60
1487	F	North Midland	41

Table 1: Phonetically Labeled Speech Times By Speaker

The training set for a speaker is composed of all the phonetically transcribed words produced by the speaker, as well as the transcriptions themselves. The testing set for each speaker is composed of the words, produced by the speaker, that do not have phonetic transcriptions. The total amount of training data for speaker 1260 (the female speaker) is in excess of 420 seconds, includes 1395 words, and comes from four different conversations. The training data for speaker 1245 (the male speaker) is approximately 180 seconds (531 words) from a single conversation. The amount of test data is much greater for both speakers: approximately 3590 seconds (16891 words) for speaker 1260 and 1360 seconds (5251 words) for speaker 1245.

Acoustic files are named according to the convention [word][number].int16 where "number" corresponds to the placement order of the word in the conversation. An example of the naming convention used can be seen in the naming of the first five words extracted from channel A of conversation sw02830: "yeah.1.int16", "i've.2.int16", "been.3.int16", "sitting.4.int16", "here.5.int16". The phonetic transcription for a word is named using the convention [word][number].phn. The "word'' and "number'' fields of a phonetic file are identical to the respective fields of the corresponding acoustic word file. For example, a pair of files in the training data set of speaker 1260 are "yeah.1.int16" (acoustic word file) and "yeah.1.phn" (phonetic transcription).

The directory structure used to store the isolated words extracted for speaker 1245 is shown in Figure 3. The directory structure for a speaker first divides the words into the training and test sets, then separates words coming from different conversations (e.g. conversation number sw02681A vs. conversation number sw02696B). For the training data, the

transcription files are further separated from the acoustic files. In Figure 3, the numbers in parentheses indicate the number of seconds of speech in each directory. For example, the extracted words from conversation sw02681A contribute about 128 seconds of speech to the data set of isolated words from speaker 1245.

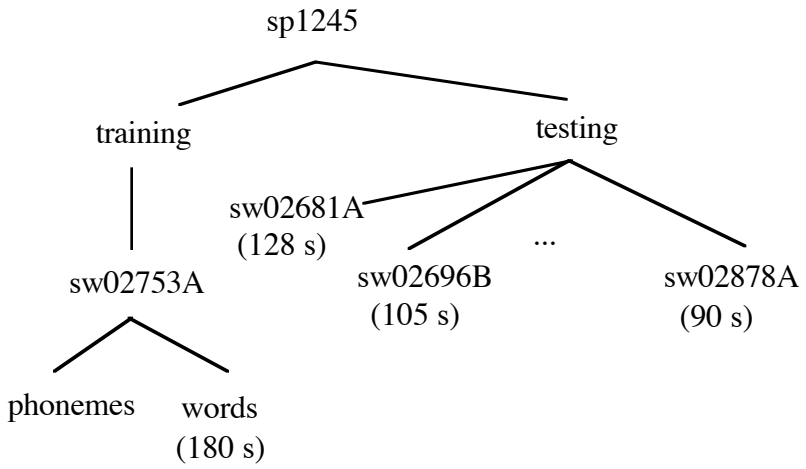


Figure 3: Directory Structure for Speaker 1245

4.1.2 Creation of Isolated-Word Data Set

In order to extract isolated words from the Switchboard data, it was necessary to locate all the conversations that included the chosen speakers, find the beginning and ending times of words, and use the beginning and ending times to extract the word acoustics and the phonetic transcriptions (when available). The description below gives additional details of how these steps were performed.

Having identified one male (speaker 1245) and one female (speaker 1260) with the most phonetically labeled speech, we wrote a routine to query the conversation table file, conv.tab, for a listing of other conversations in which speakers 1245 or 1260 participated. Speaker identities were verified by checking against a list of conversation IDs (file "chan_err.cnv") for known cases of inverted channel labels, i.e., the speaker in channel A mislabeled as being in channel B and vice' versa. Speaker identities were also checked by ear to ensure that the identity corresponding to a voice was consistent across conversations. Interestingly, we found one or two cases of channel inversion that were not listed in the "chan_err.cnv" file.

The .mrk file for a conversation contains information about the starting and ending time of each word (as well as each non-word, e.g., laughter) produced during the conversation. Using the .mrk files, it was possible to extract the beginning and ending times for only the words (not non-words) for the speaker to be studied.

The speech files included on the Switchboard CD-ROMs are stored in NIST sphere format: a 1024-byte header followed by the mu-law encoded acoustics of a two-way telephone conversation sampled at 8 kHz. We removed the header information, separated the channels, used the .mrk files to locate the beginning and ending of each word, and converted

the acoustics for each word to linearly quantized data (as opposed to mu-law encoding). Each word is stored in a separate file and named as described above.

Thanks to the ICSI, phonetic transcriptions are available for some of the conversations. The phonetic transcriptions are given in .phn files that contain the ending times (and therefore, implicitly, the beginning times) for each phone in the conversation. The .mrk files used to find the word beginning and end times for the acoustics were not used for finding the beginning and end times for the words in the phonetic transcriptions. We did not use the .mrk files because the phonetic transcriptions are only made for some portions of the conversations, and are not easily aligned with the .mrk files. Instead, the word beginning and end times found in the word level transcriptions were used for extracting phonetic transcriptions. The phonetic transcription for each word is stored in a separate file and named as described above.

4.1.3 Problems With Data Set

There are several problems with the isolated-word dataset. In this section we discuss what the problems are, how they originated (if known), and what impact (if any) the dataset might have on recognition performance.

The most serious problem is with the misplacement of phones in the phonetic transcriptions of a word. A routine was written to automate the extraction of phones for each word in the dataset. Unfortunately, our algorithm occassionally makes errors indetermining which phones in the transcription correspond to the word of interest. One of two types of errors typically occurs: extra phones are inserted in the .phn word files or legitimate phones are mistakenly left out of the .phn word files. For example, if the original transcription of "for each" is: /f o r i ch/, and we want to get the transcription of the word "for", our algorithm occasionally will give erroneous transcriptions like /f o/, /o r/, or /f o r i/.

To explain why phone errors occur, we will briefly review the format of the input files used and step through a more detailed look at the algorithm implemented. Examples of word and phone-level transcription files (sw2830A -ws96-i-0010.wrd and sw280A -ws96-i-0010.phn) are shown in Figure 4 and Figure 5. Inspection of these figures reveals the information contained in the word and phone-level transcription files. Following a header are three columns: column one gives the end time (in seconds) of the event listed in column three, while column two lists a number representing a color used for plotting purposes.

A straightforward algorithm reads the first word from the .wrd file ("YEAH"), converts to lower-case, and opens a file named yeah[number].phn. The first phone from sw2830A-ws96-i-0010.phn is read. The phone end time is compared to the end time of the word "YEAH". If the phone end time is less than or equal to the word "YEAH" end time, compute the duration time, store the phone in yeah[number].phn and get the next phone. Keep reading phones and storing in yeah[number].phn file until a phone end time is greater than the word end time. Once the phone end time is greater than the end time of "YEAH", close the file yeah[number].phn, get the next word from sw2830A-ws96-i-0010.wrd (i.e "WHAT"), open a file named what[number+1].phn and begin extracting phones as before. Repeat until all of the words in the .wrd file have been processed.

```
Signal 2830-A-0010
type 0
color  121
comment  $Locker:   $
font  -misc-*-bold-*-*-*-15-*-*-*-*-*-*-*
separator  ;
nfields  1
#
      0.549287   121 H#
      0.772359   121 YEAH
      0.949164   121 WHAT
      1.041700   121 DO
      1.159020   121 YOU
      1.477930   121 LIKE
      1.742310   121 H#
```

Figure 4: Contents of sw2830A -ws96-i-0010.wrd File

```
comment $Header: /u/stp/data/ws96to97/sri-align.1/phn/
RCS/2830-A-0010.phn ,v 1.2 1997/06/20 20:05:12 joyh Exp $
comment $Locker:  $
#
      0.549287   121 h#
      0.640000   121 y
      0.749973   121 ae
      0.772359   121 ae_cr
      0.870000   121 w_cr
      0.935945   121 ax
      0.960000   121 dx
      1.041700   121 iy
      1.080660   121 y
      1.159020   121 ux
      1.235030   121 l
      1.388700   121 ay
      1.477930   121 k
      1.742310   121 h#
```

Figure 5: Contents of sw2830A-ws96-i-0010.phn File

Using the above algorithm, we discovered that phones were being omitted. The simple reason is that the word level transcription and the phonetic transcription don't always agree on when the word starts and ends -- the end time of the last phone of the word in the phonetic transcription is not always the same as the end time of the word in the word-level transcription. Thus, phone end times greater than word end times resulted in the last phone being omitted. Alternately, if the end time of the last phone is less than the word end time, a phone can be appended to the end of the first word, and ommitted from the beginning of the next word. A slight modification, therefore, was made to implement our version of the basic algorithm. We determined empirically that extending word end times by 0.03 s appeared to minimize this phenomena. Unfortunately, this modification did not totally eliminate the problem. Phones are still occassionally inserted or deleted. It is our estimate that anywhere from 10-15% of the phonetically transcribed word files are in error. Errors of this type are

especially troublesome since they have a direct affect on our ability to train a recognition system.

There were a number of discrepancies between words listed in the .mrk file and the corresponding word-level transcription files. It became apparent when processing data that file names for acoustic data (obtained from .mrk files) did not always match the file names of phonetically transcribed data (generated from word-level transcription files). Several examples illustrating the difference between file names are shown in Figure 6.

Speaker 1260
Conversation 2830A

Acoustic Word Files	Phoneme Files
accent.1445.int16	accents.1445.phn
all.133.int16	and.133.phn
cause.181.int16	because.181.phn
jeez.1366.int16	gee.1366.phn
husband.403.int16	husband's.403.phn

Figure 6: Inconsistent File Names

Evidently, human errors were made in transcribing the data; orthographic transcriptions did not always match the word-level transcriptions. Inconsistent file names make it more difficult to reliably train a recognition system on the data.

There are also problems with the acoustics of many isolated words. A typical conversation is six minutes in duration and contains 1000-2000 words. Conversations were time-aligned with a supervised speaker-independent automatic speech recognition algorithm based on orthographic transcriptions, on-line dictionaries, and HMM phone models (Wheatley et al., 1992). While largely successful, we detected situations where words appeared to be misaligned. (these observations were made in listening to isolated words extracted from conversations). This result is really not very surprising since the majority of words are correct to within a second at least. Although many words are, in fact, correct to within one or two time frames (20 ms), there appear to be situations where the alignment is off by more than a few time frames. In conversational speech this may not present much of a problem, but in isolated-word recognition words misaligned by up to a second are much more problematic and difficult to recognize.

4.2 German

The German data set was collected during the course of dissertation work by P. Hoole at the Institute for Phonetics in Munich, Germany and is described in detail in Nix's dissertation (Nix, 1998). The data set consists of measurements of both acoustics and articulation with accompanying time-aligned phonetic transcriptions. The complete data set contains speech from 5 male speakers and 1 female speaker. Each speaker read 108 sentences. The sentences were the same for all six speakers. Each sentence is about 4 seconds long.

The acoustics were recorded using a room-placed microphone and were originally sampled with 16 bit resolution at 16 kHz. The data was downsampled to 11025 Hz for our experiments.

Time aligned phonetic labels were obtained using speaker-dependent hidden Markov models and corrected by hand when necessary.

Articulator measurements consist of measurements of the positions of small Electromagnetic Midsagittal Articulometer (Perkell et al., 1992) receiver coils glued to the tongue, jaw, and lips of the subjects. The receiver coil positions were measured with 16 bit resolution at 250 Hz. The articulator measurements were downsampled as necessary to make one articulator measurement for each acoustic frame, and were then low-pass filtered to remove energy above 15 Hz.

It took very little data to find the flaws in our first three word models. For example, when recognition performance using the second word model was very poor for a few sentences from one speaker, we looked for possible problems and found the probability/PDF height mismatch that was the root of the problem. Similarly, when the $p(\mathbf{x}|f)$ distributions for a speaker showed large overlaps, we did not need to do a full recognition test to know that the recognition performance would be poor for the third word model. Thus, not all of the data was used. More details about what was done with this data can be found in the dissertation by Nix.

5. MALCOM word model (version 1)

The goal of speech recognition is to find the most probable word given the acoustic evidence, i.e., a string of VQ codes or acoustic features. Speech recognition algorithms typically take advantage of the fact that the probability of a word, w, given a sequence of VQ codes, **c**, can be calculated from:

Equation 20
$$P(w \mid \mathbf{c}) = \frac{P(\mathbf{c} \mid w)P(w)}{P(\mathbf{c})}$$

Since the probability of a word, $P(w)$, is typically given by a language model, and the probability of a sequence of VQ codes, $P(\mathbf{c})$, is constant during recognition, our goal is to create a word model giving the probability of the sequence of VQ codes given the word, $P(\mathbf{c} \mid w)$.

In the original proposal, we suggested three routes to isolated word recognition using MALCOM. These routes, as described in the original proposal, are copied below:

1. Set the CM a priori probabilities of observing each code to the a priori probabilities of each code given the word to be modeled.
2. For each word, find a single trajectory through the CM that best accounts for the word acoustics. Trajectory scaling will be used to accommodate different speaking rates. Alternately, dynamic time warping could be used to compare trajectories.
3. Look for qualities of CM trajectories that provide invariant cues for the different phonemes.

These approaches are presented here mostly to emphasize the amount of progress made over the course of the year. The first two approaches are seriously flawed and the third approach is essentially exploration. Although we still believe some basic exploratory research should be done, it is less likely to result in improved recognition performance in the short term.

The first two approaches were more goal-oriented than the third, but both suffered from being word-based: they don't use subword models. Using word-based models is a problem for large vocabulary speech recognition, where we can't expect to have examples of each possible word for training. So, to take maximal advantage of a set of training data,

speech recognition algorithms commonly attempt to model subword units (phonemes, diphones, etc.) instead of whole words. The subword units can then be concatenated to create word models. In contrast to creating a model for each of many thousands of words, since there are only 40 or so phonemes used in English, it is much easier to get adequate training data to determine the parameters of phoneme models than to determine the parameters of word models. With models for 40 or so phonemes and a dictionary that describes each word as an ordered set of phonemes, it is possible to create word models for words that have never even been observed.

The work described below uses phoneme models. However, it might make more sense to use subphoneme units, e.g. treat /d/ as the sequence: /transition to d-closure/, /silence/, /d-burst/ ... We believe this is only one of the many alternatives that could potentially improve the performance of a MALCOM-based speech recognition system.

6. MALCOM word model (version 2)

In discussions with our mentor, it became clear that the goal of this project should be large-vocabulary speech recognition. Thus, we modified the proposal to use MALCOM to model subword units. The approach described in this Section also has some serious flaws (discussed in Section 6.3). Nonetheless, a significant amount of research time was spent implementing this approach and discovering its flaws, so we give a fairly detailed description of the idea below.

6.1 Incorporating Articulation

From the preceding section, we know that an important goal of speech recognition is to find $P(\mathbf{c} \mid w)$. Theoretically, MALCOM could find $P(\mathbf{c} \mid w)$ using the relationship:

Equation 21 $$P(\mathbf{c} \mid w) = \int P(\mathbf{c} \mid \mathbf{X}) p(\mathbf{X} \mid w) d\mathbf{X}$$

Where $\mathbf{X}$ is a path through the continuity map (CM). If $\mathbf{X}$ is forced to be smooth we retain constraints that are analogous to requiring that the articulators move smoothly. However, since it is not practical to calculate the integral over all possible $\mathbf{X}$ values, a suboptimal technique for finding $P(\mathbf{c} \mid w)$ must be employed. In such situations, one can use the maximum value of $P(\mathbf{c} \mid \mathbf{X}) p(\mathbf{X} \mid w)$ as an estimate of $P(\mathbf{c} \mid w)$. For example, given a sequence of acoustic codes and while requiring $\mathbf{X}$ to be smooth, we can find:

Equation 22 $$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} \left[P(\mathbf{c} \mid \mathbf{X}) p(\mathbf{X} \mid w) \right]$$

and then use:

Equation 23 $$\hat{P}(\mathbf{c} \mid w) = P\left(\mathbf{c} \mid \hat{\mathbf{X}}\right) p\left(\hat{\mathbf{X}} \mid w\right)$$

as our estimate of $P(\mathbf{c} \mid w)$.

One thing these equations make clear is that we should have some way of calculating $p(\mathbf{X} \mid w)$ in order to perform speech recognition using an articulatorily constrained, maximum likelihood framework for speech recognition.

Recall that there is already some evidence that the positions in a CM made using MALCOM roughly correspond to articulator positions. This suggests that the CM positions corresponding to a given phoneme will be distributed in some relatively well-specified region of the CM. For example, suppose that one dimension (call it y) of the CM corresponds to lip rounding and another dimension (call it x) corresponds to the x position of the tongue tip. Since /t/ is always produced with the tongue tip forward but with varying degrees of lip rounding, we might expect the distribution of CM positions observed during the production of /t/ to have a small variance on the x-axis but a larger variance on the y-axis.

Thus, a phoneme can be characterized by a distribution of positions in the CM. Doing so is simple given phonetically labeled data. We simply 1) create a continuity map from the speech sounds; 2) use the MALCOM equations to find the most likely smooth paths through the CM given the acoustics; and 3) for each phoneme, find the Gaussian that maximizes the likelihood of the CM positions.

Using $p(\mathbf{x} \mid f_i)$ to denote the distribution of positions in the continuity map given the phoneme f_i, and assuming conditional independence, we can write:

Equation 24
$$p(\mathbf{X} \mid w) = \prod_i p\left[\mathbf{x}(t) \mid f(t)\right]$$

6.2 Aligning text to speech

Note that describing a word as an ordered sequence of phonemes is incomplete in that it leaves out information about the relative timing of the phonemes. This is a problem because it is impossible to determine if $\mathbf{X}$ is a smooth path if we only know that $\mathbf{x}_1$ occurs before $\mathbf{x}_2$, $\mathbf{x}_2$ occurs before $\mathbf{x}_3$, etc. In order to fully characterize a particular production of a word we need to say not only what phonemes were produced, but when the phonemes were produced. However, it would not be a good idea to have a word model deterministically specify when the phonemes occur, because the timing of the phonemes will vary with speaking rate, emphasis, etc. Thus, in order to find the word that maximizes the probability of a sequence of VQ codes, we should use a probabilistic algorithm to find the alignment of the phonemes with the acoustics and to find the smooth path that, together with the alignment, jointly maximizes the probability of the VQ code sequence. The process outlined below was intended to give 1) the probability of the acoustic sequence given a word, and 2) the alignment of the phoneme centers with the acoustics. As such, it is similar to using the Viterbi algorithm to align text with speech.

Putting the problem into equation form, we need to first find:

Equation 25
$$\begin{aligned}\left[\hat{\mathbf{X}}, \hat{\mathbf{t}}\right] &= \arg\max_{[\mathbf{X},\mathbf{t}]}\left[P(\mathbf{c} \mid \mathbf{X}, \mathbf{t})p(\mathbf{X} \mid w, \mathbf{t})p(\mathbf{t})\right] \\ &= \arg\max_{[\mathbf{X},\mathbf{t}]}\left[P(\mathbf{c} \mid \mathbf{X})p(\mathbf{X} \mid w, \mathbf{t})p(\mathbf{t})\right]\end{aligned}$$

where

Equation 26
$$\mathbf{t} = \left[\Delta t_1 \quad \Delta t_2 \quad \cdots \quad \Delta t_M\right]$$

and Δt_i is the time between the center of phoneme *i-1* and phoneme *i* (the 0^{th} phoneme is the beginning of the word). Then we can use:

Equation 27 $\hat{P}(\mathbf{c} \mid w) = P\left(\mathbf{c} \mid \hat{\mathbf{X}}\right) p\left(\hat{\mathbf{X}} \mid w, \hat{\mathbf{t}}\right) p(\hat{\mathbf{t}})$

to estimate $P(\mathbf{c} \mid w)$.

It is in $p(\hat{\mathbf{t}})$ that we could add information about how speaking rate transforms the relative timing of the phonemes. For example, we could make $p(\hat{\mathbf{t}})$ a function of speaking rate and constrain speaking rate to vary slowly over time. In addition, since $p(\hat{\mathbf{t}})$ varies by word, we either have to specify $p(\hat{\mathbf{t}})$ for each word (increasing the complexity of the word model) or we need to find a way to determine $p(\hat{\mathbf{t}})$ just from knowing the phoneme sequence. It would be possible to find $p(\hat{\mathbf{t}})$ from the phoneme sequence if, for example, the time between /a/ and /b/ is relatively independent of the word in which /a/ and /b/ are observed.

To review, $\hat{\mathbf{t}}$ gives the best alignment of the phoneme centers with the speech. Given $\hat{\mathbf{t}}$, it is possible to find the $\hat{\mathbf{X}}$ in Equation 22 using a maximization algorithm such as conjugate gradient ascent. Finally, $\hat{\mathbf{X}}$ is used in Equation 23 to get an estimate of the likelihood of the acoustic sequence given the word. The likelihood of the acoustics sequence given the word can then be used with a language model to perform speech recognition.

6.3 Flaws

In our revised proposal, we wrote: " it will be probably be necessary to specify a $\hat{\mathbf{t}}$, find $\hat{\mathbf{X}}$, record the value of $\hat{P}(\mathbf{c} \mid w)$, and repeat this process with different values of $\hat{\mathbf{t}}$. The $\hat{\mathbf{t}}$ for which $\hat{P}(\mathbf{c} \mid w)$ is maximized will be chosen."

Finding the optimal components of $\hat{\mathbf{t}}$ is very computationally expensive. If we only know the word exists somewhere within N time steps, then an exhaustive search of $\hat{\mathbf{t}}$ values require O(N!) calculations of the probability of the VQ code sequence given the alignment -- a calculation that is itself very time consuming.

We implemented a highly constrained search of possible alignments that can probably be improved upon. However, even if an exhaustive search was performed, it is now clear that this approach to word modeling would not yield good recognition performance. The main flaw of this approach to word modeling is that $P(\mathbf{c} \mid \mathbf{X})$ is a real probability (is between 0 and 1 and sums to 1) whereas $p(\mathbf{X} \mid w)$ is the height of a probability density function. This is a problem because the effect of $P(\mathbf{c} \mid \mathbf{X})$ on $P(\mathbf{c} \mid w)$ is minimal compared to the effect of $p(\mathbf{X} \mid w)$. Essentially, the algorithm pays very little attention to whether the model actually predicts the acoustics. This was true to such an extent that in preliminary studies of the German data set, the algorithm could not correctly discriminate between "das" and "der".

This flaw prompted us to search for a word modeling approach that does not require multiplying probabilities by heights of density functions. The first such approach is described in the next section.

7. MALCOM word model (version 3)

An assumption spelled out in the preceding Section is that positions in a MALCOM-derived continuity map correspond to articulator positions, and therefore should correspond, to some extent, to phonemes. As also mentioned previously, it may well be the case that the positions do not correspond to phonemes per se, but instead correspond to subphonetic

units like bursts or closures. Nonetheless, a good starting point is to assume that phoneme identity can be determined from static articulator positions, and presumably, from continuity map positions.

In the third word model we retain the idea that we can model the probability of a continuity map position given a phoneme by a Gaussian PDF. Using $p(\mathbf{x} \mid f_i)$ to denote the Gaussian distribution of positions in the continuity map given the phoneme f_i, we can write the conditional probability that the phoneme at time t is f_i:

Equation 28
$$P\left[f(t) = f_i \middle| \mathbf{x}(t)\right] = \frac{p\left[\mathbf{x}(t) \mid f_i\right] P(f_i)}{\sum_j p\left[\mathbf{x}(t) \mid f_j\right] P(f_j)}$$

It can easily be verified that $P\left[f(t) = f_i \middle| \mathbf{x}(t)\right]$ is a probability -- it is always between 0 and 1 and the sum over *i* is 1. This equation, then, allows us to calculate the probability of each phoneme for each window of acoustics, given a path through the continuity map.

The basic idea of the third word model is to use $p\left[f(t) = f_i \middle| \mathbf{x}(t)\right]$ in conjunction with the probability of a phoneme sequence given a word, $P\left[f(t) = f_i \middle| w\right]$, to get an estimate of the probability of a word. Since a variety of standard techniques can be used to create a Markov model giving $P\left[f(t) = f_i \middle| w\right]$, we will assume that this probability can be found and give it no further discussion until we present the actual recognizer implementation in Section 9.3.

7.1 Combining the word model with a lattice

One advantage this word model has over the previous models is that it can be combined with the kinds of lattice structures commonly used in speech recognition. To see how, define variables reminiscent of the HMM forward algorithm:

Equation 29
$$a_{ij} = P\left[f(t) = f_i \middle| f(t-1) = f_j, w\right]$$

Equation 30
$$b_i(t) = P\left[f(t) = f_i \middle| x(t)\right]$$

Equation 31
$$\alpha_i = P\left[f(1) = f_i \middle| w\right]$$

Equation 32
$$\alpha_i(t) = P\left[f(t) = f_j \middle| w, x(t), x(t-1), x(t-2), \ldots, x(1)\right]$$

The reader can confirm that

Equation 33
$$\alpha_i(1) = b_i(1)\alpha_i$$

and

Equation 34
$$\alpha_i(t) = b_i(t)\sum_j a_{ij}\alpha_j(t-1)$$

Using these recursively calculated probabilities we can find

Equation 35 $$P[w|\mathbf{X}] = P[w|x(t), x(t-1), x(t-2), ..., x(1)] = \sum_i \alpha_i(t) P(w)$$

Ideally, we would find

Equation 36 $$P[w|\mathbf{c}] \propto \int P[w|\mathbf{X}] P[\mathbf{X}|\mathbf{c}] d\mathbf{X}$$

but since calculating this integral is not practical, we will assume that $P[\mathbf{X}|\mathbf{c}]$ is only non-zero in an infinitesimal region around

Equation 37 $$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} P[\mathbf{c}|\mathbf{X}]$$

and calculate

Equation 38 $$P[w|\mathbf{c}] \propto P[w|\hat{\mathbf{X}}]$$

The basic idea, then, is to start by finding the smooth path through the continuity map that maximizes the probability of the VQ code sequence. Then use that path to get the probability of each phoneme for each acoustic window. Then combine the probabilities of each phoneme given the path, with the phoneme probabilities given the word, and the prior word probability, to get an estimate of the posterior probability of the word.

7.2 Flaws

The basic approach is mostly sound, but positions in the continuity map give insufficient information about phoneme identity. This assertion was demonstrated using the German data set. Continuity maps were created for a subject in the German data set and then the $p(\mathbf{x} \mid f_i)$ Gaussians were calculated. Unfortunately, the variances of these Gaussians were large compared to the distances between the means of the Gaussians -- implying that positions in the CM give very little information about the phoneme identity.

If the positions in the continuity map are not good at discriminating between phonemes, then we have little chance of getting good recognition performance. However, the next section shows how to modify the MALCOM algorithm to force the continuity map positions to give information about phoneme identity.

8. MALCOM word model (Final)

The MALCOM algorithm does not use information about phoneme identity, so perhaps it should not be surprising that continuity map positions give very little information about phoneme identity. Our goal, then, is to modify the MALCOM algorithm to use phonetic labels in the training process. We essentially have two observable sequences of categorical variables available for training: the sequence of phonemes and the sequence of VQ codes. Although MALCOM was only designed to work with one sequence of observable variables, MALCOM can be modified to be able to maximize the probability of two (or more) sequences of variables. Doing so forces positions in the continuity map to give information about phoneme identity as well as VQ code identity, which should lead to better speech recognition. This realization lead to the creation of Multiple-Observable MALCOM (MO-MALCOM), described below.

8.1 Multiple-Observable MALCOM training

Recall that MALCOM creates a continuity map to maximize the probability of a sequence of VQ codes given the optimum smooth path. MO-MALCOM is very similar, but it creates a continuity map that maximizes the probability of a sequence of VQ codes AND ("AND" is used in the probabilistic sense) the sequence of phonemes. The essential equation underlying MO-MALCOM is:

Equation 39 $$P[\mathbf{f},\mathbf{c}|\mathbf{X},\Theta] = \prod_t P[f(t)|\mathbf{x}(t),\Theta]P[c(t)|\mathbf{x}(t),\Theta]$$

This equation is almost identical to the MALCOM equation except that the term $P[f(t)|\mathbf{x}(t),\Theta]$ has been inserted.

The MO-MALCOM-based continuity map (defined by the parameters Θ) can be created using virtually the same steps as used to create a MALCOM-based continuity map: start with a continuity map then iteratively 1) find the smooth paths that maximize the probability of the data, 2) find the continuity map that maximizes the probability of the data given the paths, and 3) impose any necessary additional constraints on the continuity map.

8.1.1 Another simplification

Note that both the $P[f(t)|\mathbf{x}(t),\Theta]$ and the $P[c(t)|\mathbf{x}(t),\Theta]$ term are calculated using Bayes' law, and therefore both require an estimate of $p[\mathbf{x}(t)|\Theta]$. In order to make sure that both $P[f(t)|\mathbf{x}(t),\Theta]$ and $P[c(t)|\mathbf{x}(t),\Theta]$ are always between 0 and 1, we need to use $\prod_i p[\mathbf{x}(t)|c_i,\Theta]$ as our estimate of $p[\mathbf{x}(t)|\Theta]$ when calculating $P[c(t)|\mathbf{x}(t),\Theta]$, and need to use $\prod_i p[\mathbf{x}(t)|f_i,\Theta]$ as our estimate of $p[\mathbf{x}(t)|\Theta]$ when calculating $P[f(t)|\mathbf{x}(t),\Theta]$.

However, since the number of phones is much smaller than the number of VQ codes, calculating $p[\mathbf{x}(t)|\Theta]$ using $\prod_i p[\mathbf{x}(t)|f_i,\Theta]$ is much faster than calculating $p[\mathbf{x}(t)|\Theta]$ using $\prod_i p[\mathbf{x}(t)|c_i,\Theta]$. For example, if we use around 2000 VQ codes and have about 40 phones, it is about 50 times faster to calculate $p[\mathbf{x}(t)|\Theta]$ using $\prod_i p[\mathbf{x}(t)|f_i,\Theta]$. Computation time was an issue in our recognition work, so we chose to do the faster calculation for both $P[f(t)|\mathbf{x}(t),\Theta]$ and $P[c(t)|\mathbf{x}(t),\Theta]$, even though there is some chance that doing so caused some inaccuracy. It would be interesting to see what effect, if any, this decision has on recognition results.

8.2 Multiple-Observable MALCOM recognition

While the phoneme sequence is available during training, it is not available during recognition (otherwise, we wouldn't need to do recognition). This forces recognition using a MO-MALCOM continuity map to look like recognition using a MALCOM continuity map: 1) for each speech segment to be recognized, find the path that maximizes the probability of the VQ code sequence; 2) given the path, find the probability of each phoneme for each acoustic window, 3) use a word lattice as described in Section 7.1 to get an estimate of the probability of each word; 4) select the most probable word.

This version of word model was used for recognition on the Switchboard data set. The details of the implementation are given in Section 9.2.

8.3 Discriminating phonemes with MALCOM and MO-MALCOM

Continuity map positions inferred using MO-MALCOM give much more information about phoneme identity than MALCOM. A detailed description of this difference is included in Chapter 11 of a Ph. D. dissertation written by David Nix (Nix, 1998), and funded, in part, through this project.

Since the results described in Nix's dissertation are important, we will give a shortened description here. The German data were used to determine how well the pseudo-articulator positions inferred using MALCOM and MO-MALCOM discriminate phonemes. Two speakers who were not used for tuning any model parameters were used to evaluate the phoneme discrimination power. A jackknife procedure was used on the data from these two subjects: the 108 sentences were divided into 12 groups of 9 sentences. The MALCOM and MO-MALCOM algorithms were trained on 11 of the groups of sentences and tested on the remaining group. The training sets were used to estimate the $p(\mathbf{x}|f)$ PDFs using the techniques described above. After estimating distributions on the training data, continuity map paths were found using only acoustics (no phoneme labels) on the testing data.

The phoneme discrimination power of MALCOM and MO-MALCOM was also compared to the discrimination power of the actual articulator measurements, i.e., the positions of the EMMA receiver coils glued to the tongue, jaw, and lips of the subjects. To determine the phoneme discrimination ability of the measured articulator positions, the articulator positions were projected into a seven-dimensional principal component representation. The principal component subspace captures virtually all of the variance in the articulator data. Then Gaussians were found that estimated $p(\mathbf{x}|f)$ for each phoneme, where $\mathbf{x}$ is a position in the principal components space.

For each pair of phonemes, Fisher's discriminant analysis was used to find dimension of the map (either the continuity map or the principal components space) that best discriminated the phoneme pair, where the phonemes in a pair are designated f_1 and f_2. Along this best dimension, the percentage of area in common between $p(\mathbf{x}|f_1)$ and $p(\mathbf{x}|f_2)$ was computed. To the extent that this is a low value, the articulator positions (or pseudo-articulator positions) give a lot of information about phoneme identity. This technique for measuring the discrimination power of articulator positions was used for measured articulator data as well as MALCOM and MO-MALCOM estimates of articulator positions.

While we might expect that articulator measurements could discriminate phonemes with 100% accuracy, three facts prevented this from being the case. First, the articulator data do not contain measurements of velum position or voicing. Second, since every acoustic frame has a corresponding phonetic transcription, some of the frames that should be considered part of the transition between two phonemes will be arbitrarily labeled as being one phoneme or the other, causing an apparent loss of discrimination power. Third, the $p(\mathbf{x}|f)$ PDFs were estimated using Gaussians, which may not be the ideal PDFs to use.

In fact, measured articulator positions are only moderately useful for discriminating between phonemes. As should be expected, for measured articulator positions, most of the discrimination problems are between consonants that differ in voicing and nasality. For example the phonemes [p, b, m] are very difficult to discriminate (overlaps between 40% and 63% occur) based on the measured articulator data. Similarly, [t, d, n, and l] are hard to discriminate.

On the other hand, articulator positions are not very useful for discriminating between phonemes even in some cases where the phonemes differ in features other than nasality and voicing. For example, the difference between [f] and [p] is not a difference in nasality or in voicing. [f] differs from [p] in place of articulation (labio-dental vs. bilabial) and manner of articulation (fricative vs. stop). However, it is very is very difficult to discriminate between [f] and [p] (41% overlap) using articulator measurements. Similarly, [b] and [w] are difficult to discriminate using measured articulator positions (37% overlap) despite the fact that differences between these phonemes are not in the nasality or voicing features. Vowels are another example where articulator positions are only moderately useful for discrimination, where overlaps of 20% or more are not uncommon.

In the cases where articulation does not discriminate phonemes well, it would clearly be useful to use acoustics in conjunction with articulation measurements to discriminate the phonemes. This is what MO-MALCOM is apparently doing when it successfully discriminates phonemes that differ in place of articulation and also discriminates phonemes that differ in acoustic qualities. The ability of MO-MALCOM to differentiate between phonemes differing in place is demonstrated by two examples: 1) the largest overlap in MO-MALCOM PDFs between phoneme pairs composed of [p], [t], and [k] is 1%; 2) the largest overlap between phoneme pairs composed of [b], [d], and [g] is 6%. The ability of MO-MALCOM to discriminate between phonemes with similar articulation but different acoustics is also evident -- [b] and [p] have an overlap of less than 0.5%, [d] and [t] have an overlap of 2%, [k] and [g] have an overlap of 6%. Even [b] and [w], which articulator positions did not discriminate well, are discriminated well by MO-MALCOM positions (the overlap is less than 0.5%). Furthermore, MO-MALCOM continuity map positions are much better at discriminating vowels than articulator positions -- the largest overlap for MO-MALCOM is 3% and only 6 vowel pairs have overlaps larger that 0.5%. In contrast, 63 vowel pairs have overlaps of more than 0.5% using articulator data, and 9 of those vowel pairs have overlaps of more than 20%

MO-MALCOM continuity map positions also discriminate phoneme identities more accurately than MALCOM continuity map positions. The most difficult pair of phonemes for MO-MALCOM to discriminate are [r] and [l], which have 19% overlap. The next most difficult pair is [r] and the glottal stop with a 17% overlap. The vast majority of phoneme pairs have less than a 0.5% overlap and only 7 phoneme pairs have overlaps of more than 10%. While MALCOM does somewhat better at discriminating a few phoneme pairs, in general it does worse, with some overlaps as high as 29% or 34%.

Clearly, MO-MALCOM is using both acoustic and the equivalent of articulatory information in discriminating phoneme pairs, and is doing so with a high degree of accuracy, particularly considering the small training set size.

8.4 Future extensions

So far, the name Multiple-Observable MALCOM is not completely deserved. In fact, in Nix's dissertation the name Two-Observable MALCOM is used. After all, we have only described how to use MALCOM with two observable variables -- VQ codes and phonemes. However, it would be a simple extension to use this version of MALCOM with more than two observable variables. To use more than two variables we would simple insert more terms into the MO-MALCOM equation in exactly the same way that we inserted the $P(f|\mathbf{x})$ term to use phonemes as a variable. Using more observables it would be possible to have variables representing phonetic features such as "voiced" vs. "unvoiced", "Nasal" vs. "Not Nasal", etc.

The ability to use distinctive features as input to MO-MALCOM has some very real pragmatic advantages. If we were to use binary phonetic features as the categorical

variables, we would have much more training data for each feature than we have for each phoneme -- possibly allowing better recognition from less data. Training and recognition should speed up significantly as well, since calculating the probability of an item being from one of two classes is faster than calculating the probability that an item is from one of 40 classes.

Using distinctive features as input to MO-MALCOM also has some attractive theoretical aspects. Consider that using phonetic features as inputs would force the CM to arrange itself, as much as possible, into a space where the axes correspond to phonetic features, making it easier to interpret the continuity map. It might even be possible to unite some phonological theories (e.g. phonological tiers or articulatory phonology) with stochastic speech recognition algorithms. Thus, the potential is here for ultimately bridging the gap between our knowledge of speech production and our speech recognition algorithms. It is our hope that this potential can be explored in future years.

9. Baseline recognition system

The focus of this research was to examine a potential replacement for HMMs for speech recognition. Over the course of the year, we made significant departures from the original proposal, e.g., we examined three potential replacements instead of 1. We also did not expect to find a word model compatible with state-of-the-art speech recognition systems . Because of the extra time spent on finding a word model, and because we did not expect to need to duplicate components found in state-of-the-art speech recognition systems, our baseline model leaves plenty of room for improvement. Nonetheless, we did implement a basic version of a speech recognition system to help demonstrate that MALCOM-based word models can be used in place of HMM-based word models.

9.1 Signal Processing

12 LPC cepstrum coefficients and the log of the gain were extracted from 75% overlapping, 20ms windows of speech. A k-means clustering was run on the training data to find 1024 centroids for vector quantizing speech. Note that most speech recognizers subtract the cepstral mean in order to correct for changes in telephone microphones. Cepstral mean subtraction was not used in our baseline model, but would probably improve recognition performance. This would be an easy feature to add in the future.

9.2 MO-MALCOM

The MO-MALCOM algorithm was run on the VQ codes and phonemes in the training set. The $p(\mathbf{x}|c)$ PDFs were constrained to be symmetric Gaussians characterized by a mean vector and a scalar variance. The $p(\mathbf{x}|f)$ PDFs were constrained to be Gaussians with diagonal covariance matrices.

A continuity map with 7 dimensions and a cut-off frequency of 8Hz. was created for the male speaker. This solution was used because it performed well on the phoneme discrimination task described in Section 8.3. We can not be sure that is the optimal continuity map for recognition based on the phoneme discrimination results. Future tests should include varying the dimensionality and cut-off frequency to explore the effect of these parameters on word recognition.

9.3 Word Models

The baseline dictionary was derived from the dictionary of phonetically transcribed words produced at ICSI and delivered with the phonetic transcriptions in the file

"lexicon_v1_htk.text". This dictionary contains 3,567 words. The ICSI dictionary also contains homonyms. Since we are doing isolated word recognition, no language model was used, and therefore we had no chance of distinguishing words with different spellings but the same pronunciations. Thus, only words that did not have an identical pronunciation somewhere earlier in the dictionary were kept, eliminating 163 words from the dictionary. It was also noticed that the male speaker's training set contained no examples of the phoneme "oy". Since phonemes that do not occur in the training set can not be recognized, we eliminated all words in the lexicon that contained the phoneme "oy" -- eliminating another 59 words. The final, "baseline", dictionary contained 3,345 words.

Note that the baseline dictionary contained only canonical pronunciations of words, e.g., the word "about" is given the single pronunciation [ax b aw t]. The pronunciation [em b aw t], which actually occurs in the training set, is not included in the dictionary. At this point, we are not yet sure that the dictionary contains all the words (excluding homonyms) that occur in the testing set, but it seems unlikely. Clearly, the dictionary could be much improved -- with a corresponding improvement in recognition performance.

For each word, each pronunciation that occurred in the dictionary was represented by a lattice with as many states as there are phonemes in the word. Each state in the lattice represented a phoneme. Transitions that skipped one or more phonemes were not allowed -- only transitions from a phoneme to itself and transitions to the next phoneme were allowed. Thus, the word pronounced [s ih t] is given a lattice in which there is a transition from the [s] to the [i], and from [i] to [t], as well as transitions from [s] to [s], [i] to [i], and [t] to [t]. The values of the transition probabilities were derived from the average duration of each phoneme. That is, we calculated the average duration, in acoustic frames, of each phoneme. Then we set the transition probability of a phoneme state to itself so that the average number of acoustic frames spent in a phoneme state was equal to the average duration of the phoneme. Since there were only two transitions from each state (either to itself or to the next phoneme) the transition to the next phoneme could be easily calculated as 1 minus the probability of making a transition to self. These lattices are overly simple for the problem, but were the best that we could produce in a limited time. We would suggest that instead of having LANL recreate word lattices like those that have been created at several other sites, duplicating effort, it would be better, if possible, to do some sort of collaborative effort in future years.

As detailed in earlier derivations, we must have an estimate of the a'priori probability of a word in order to estimate the a'posteriori probability of the word. Since the training set contained very few words relative to the testing set, we could not use prior probabilities estimated from the training set on the testing set. Doing so would give us words with a'priori probabilities of 0 -- never allowing them to be recognized. Thus, we gave each word the a'priori probability of 1 divided by the number of words in the dictionary. In all likelihood, better estimates of the a'priori probabilities of the words would greatly improve recognition performance.

10. Results and discussion

Due to the incredibly long running time of our MATLAB version of the MO-MALCOM training and recognition algorithms, and the fact that the main programmer for the project left LANL near the beginning of the final month of this project, the MALCOM speech recognition system has still not been run on the testing set. From the programmer who left, we have only received preliminary results -- the result of running the recognition algorithm on the first 350 words in the male speaker's training set. For this data, we are currently achieving 40% word recognition. Clearly this is an overestimate of the actual recognition performance, which should be measured on a testing set never used for training or

exploratory analyses. We have developed a large body of C++ code that we expect will significantly improve the running time of our algorithms. We are currently funded to put the MALCOM algorithm onto a DSP chip, which also opens the possibility of doing MALCOM-based speech recognition on DSP chips in subsequent years.

The real question to be answered is whether the acoustic modeling approach we developed over the last year will improve recognition performance when added to a state-of-the-art system. Unfortunately, this question is impossible to answer at this point. There are two facts that make it impossible to answer this question: 1) Our focus on acoustic modeling made it impossible to create a state-of-the-art system in which to embed the acoustic modeling, and 2) the data set we used is so different from any previously used data that it is even difficult to determine how well any other system would perform on our data.

While it is unfortunate that we can not answer the question of whether the MALCOM approach will improve recognition performance, it is no surprise that this question can not be answered. LANL did not expect, nor did we ever suggest that a state-of-the-art system could be created with the efforts of 2 half-time staff members and a graduate student in the course of a year, particularly when we were not going to use HMMs -- the core of most modern speech recognition systems, and a core that took 30 years to develop to its current state. We are, in fact, pleased to have accomplished as much as we did and are proud of this work.

It is also no surprise that there are difficulties in comparing our performance on this data set to any other data set. Our sponsors were well aware of the fact that our data set is like no other data set, but weighed this disadvantage against the advantages of gaining experience of working with extremely complex data. In the end, our sponsors chose to have us work with the Switchboard data, a fact that will undoubtedly aid in performing the research we will be conducting over the next year.

However, it is worth summarizing factors that, if mitigated, would improve recognition performance. We can expect improved performance if we 1) adjust the signal processing to incorporate cepstral mean subtraction and variance normalization; 2) use a dictionary that contains pronunciations that actually occur in casual speech; 3) use better estimates of the prior word probabilities; 4) use much more training data; 5) use more sophisticated word lattices, and 6) attempt to combine different recognition systems; and 7) adjust the dimensions and cut-off frequency of the continuity map to optimize recognition performance.

Current recognition results notwithstanding, the phoneme discrimination results described in Section 8.3, which starts on page 34, strongly suggest that MO-MALCOM is capable of giving excellent recognition results. This suggests to us that further work should be done to improve the various components of the recognition system that necessarily were given little attention over the course of the year. When these additional components are brought up to state-of-the-art quality, we believe that MO-MALCOM will be shown to enable excellent recognition performance. It is our (admittedly biased) opinion that MALCOM is among the most innovative and promising technologies used for speech recognition. While it should be expected that several years of work will need to be expended to improve speech recognition using MALCOM, it has an excellent chance of dramatically increasing recognition performance.

11. Lagniappe: Speech compression

In Section 3.3.2.2 it was noted that two parameters of MALCOM, the dimensionality of the continuity map and the cut-off frequency, could be adjusted to minimize the number of bits required to transmit the data. The obvious implication is that MALCOM should be studied

for its potential to compress speech. While this was not the primary goal of the project, we did obtain some preliminary results on speech compression that should be of interest.

We arbitrarily chose one of the male speakers from the German data set (described in Section 4.2 starting on page 26) as a subject. After removing silence from the speaker's acoustics, the acoustics were processed using 75% overlapping 23ms Hamming windows. Each window of speech was normalized to have zero mean and a variance of 1, and then 32 cepstrum coefficients were calculated to create one 32-dimensional feature vector for each window of acoustics.

Using the first 90 sentences produced by the subject (approximately 4 minutes of speech), a 512-code vector quantization codebook was created. The codebook was used to quantize all of the subject's speech -- the first 90 sentences and the remaining 18 sentences.

Continuity maps with varying dimensionality and cut-off frequency were created. the dimensionality of the continuity maps varied between 3 and 7. Cut-off frequency values of 3, 4, 5, 7, and 10 Hz. were used. For each map, the number of bits required to send the training data was computed. The map that required the fewest number of bits to send the training data had 4 dimensions and a 3 Hz cut-off frequency. This map was used on the testing data -- the 18 sentences that were not used for creating the VQ codebook or for training MALCOM.

The test data for this subject consists of about 52 seconds of speech and was encoded using 9025 VQ codes. Thus, it would typically take 9025codes*9bits/code = 81225 bits to transmit the VQ codes -- a highly compressed version of the speech. If we take advantage of the fact that different VQ codes have slightly different occurrence probabilities, then we could theoretically use Huffman coding (Sayood, 1996) to transmit the VQ codes at the entropy rate of 75863 bits -- a 7% reduction.

To estimate the number of bits needed to transmit the speech using MALCOM, assume that the smooth paths calculated by MALCOM can be transmitted at 5 bits/sample, which is a reasonable estimate because that is the number of bits/sample required to transmit LPC codes in the U.S. Standard LPC-10 speech compression system. If the assumption holds, then the smooth paths calculated by MALCOM could be transmitted using 6297 bits (5 bits/sample * 1 sample/dimension * 4 dimensions * 6 samples/second * ~52 seconds), which is a 92% reduction in bits.

However, it would not be possible to recover the VQ codes losslessly if we only transmitted the smooth paths. Therefore, for lossless transmission of the VQ codes, we need to add the number of bits required to transmit the VQ codes given the smooth paths, i.e. the error bits. The number of error bits turns out to be 54,842. Thus, transmitting the paths and the error bits would require 61139 bits, which is a 25% reduction in the number of bits need to transmit the VQ codes.

Thinking optimistically, it may be possible to improve on this compression rate if we can transmit some dimensions with fewer bits per sample, or at a different sampling rate, or if we are willing to accept some loss. Thinking pessimistically, however, the entropy rate is a theoretical minimum that is typically not achievable in practice. Nonetheless, these results strongly suggest that more work should be put into this approach to speech compression.

12. Bibliography

Deng, L., & Sun, D. (1994). A statistical approach to automatic speech recognition using the atomic speech units constructed from overlapping articulatory features. Journal of the Acoustical Society of America, 95(5), 2702-2719.

Erler, K., & Deng, L. (1992). HMM representation of quantized articulatory features for recognition of highly confusable words. ICASSP, 1, 545-548.

Gray, R. (1984). Vector Quantization. IEEE Acoustics, Speech, and Signal Processing Magazine, 4-29.

Hogden, J. (1996). Improving on hidden Markov models: An articulatorily constrained, maximum likelihood approach to speech recognition and speech coding (unclassified LA-UR-96-3945). Los Alamos, NM: Los Alamos National Laboratory.

Hogden, J., Zlokarnik, I., Lofqvist, A., Gracco, V., Rubin, P., & Saltzman, E. (1996). Accurate recovery of articulator positions from acoustics -- new conclusions based on human data. Journal of the Acoustical Society of America, 100(3).

Huang, X. D., Ariki, Y., & Jack, M. (1990). Hidden Markov Models for Speech Recognition. Edinburgh: Edinburgh University Press.

Lee, K. F. (1989). Automatic Speech Recognition: The Development of the SPHINX System. Boston: Kluwer Academic Publishers.

Markel, J., & Gray, A. (1976). Linear Prediction of Speech. New York: Springer-Verlag.

McGowan, R., & Faber, A. (1996). Introduction to papers on speech recognition and perception from an articulatory view. Journal of the Acoustical Society of America, 99(3), 1680-1682.

McGowan, R., & Lee, M. (1996). Task dynamic and articulatory recovery of lip and velar approximations under model mismatch conditions. Journal of the Acoustical Society of America, 99(1), 595-608.

Muller, E., & McLeod, G. (1982). Perioral biomechanics and its relation to labial motor control. Journal of the Acoustical Society of America, 78(Suppl. 1), S38.

Nelson, W. (1977). Articulatory feature analysis -- I. Initial processing considerations. Memorandum, Bell Laboratories.

Nix, D. (1998). Machine learning methods for inferring vocal-tract articulation from speech acoustics. Unpublished Ph.D., University of Colorado, Boulder, CO.

Papcun, G., Hotchberg, J., Thomas, T., Laroche, F., Zacks, J., & Levy, S. (1992). Inferring articulation and recognizing gestures from acoustics with a neural network trained on X-ray microbeam data. Journal of the Acoustical Society of America, 92(2), 688-700.

Perkell, J., Cohen, M., Svirsky, M., Matthies, M., Garabieta, I., & Jackson, M. (1992). Electromagnetic midsagittal articulometer systems for transducing speech articulatory movements. Journal of the Acoustical Society of America, 92(6), 3078-3096.

Rose, R., Schroeter, J., & Sondhi, M. (1996). The potential role of speech production models in automatic speech recognition. Journal of the Acoustical Society of America, 99(3), 1699-1709.

Roweis, S. (1998). , .

Sayood, K. (1996). Introduction to Data Compression. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Schroeter, J., & Sondhi, M. (1994). Techniques for estimating vocal-tract shapes from the speech signal. IEEE Transactions on Speech and Audio Processing, 2(1), 133-150.

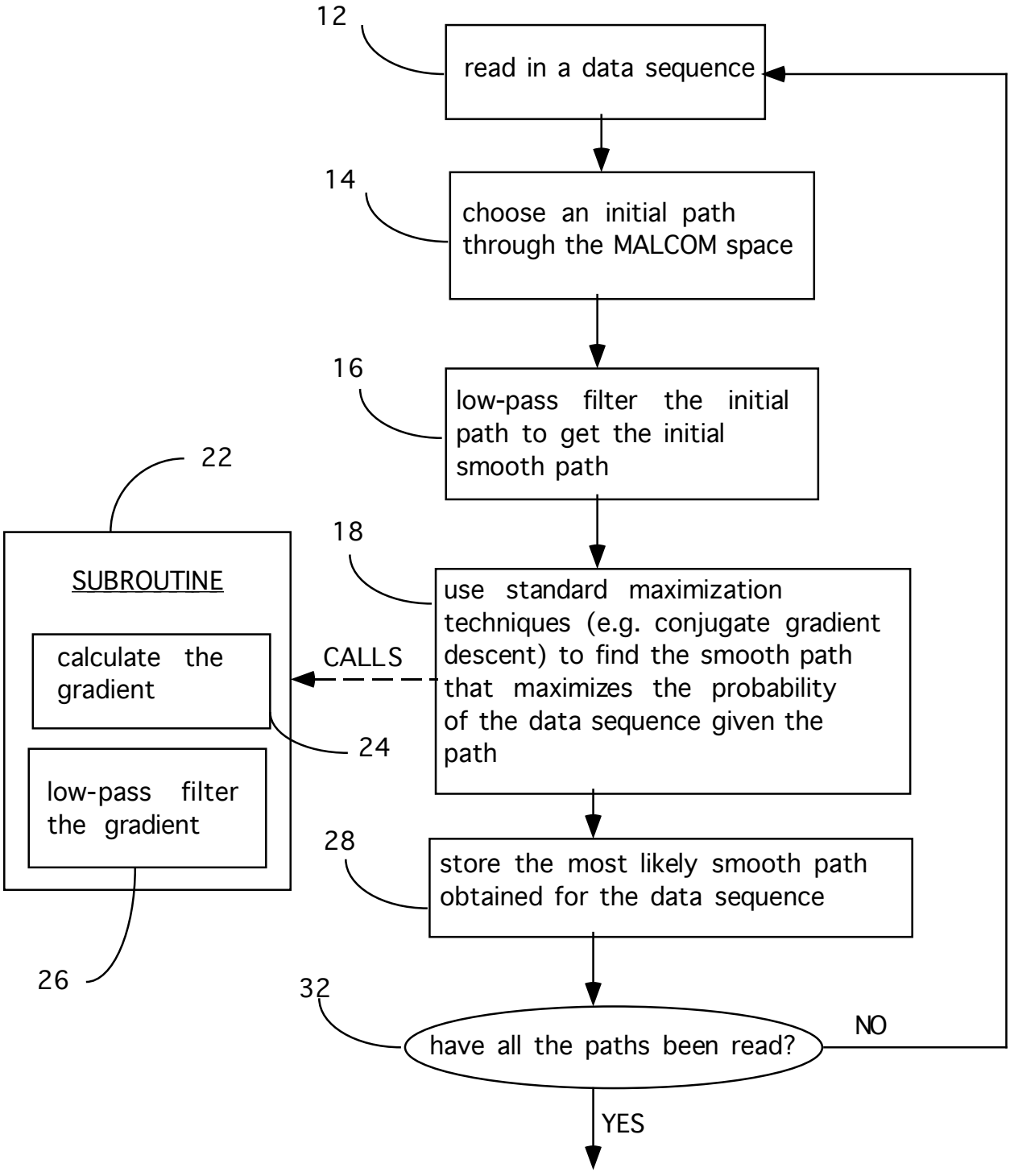
Sondhi, M. (1979). Estimation of vocal tract areas: the need for acoustical measurements. IEEE trans. ASSP, 27(3), 268-273.

Wakita, H., & Gray, A. (1975). Numerical determination of the lip impedance and vocal tract area functions. IEEE trans. ASSP, 23(6), 574-580.

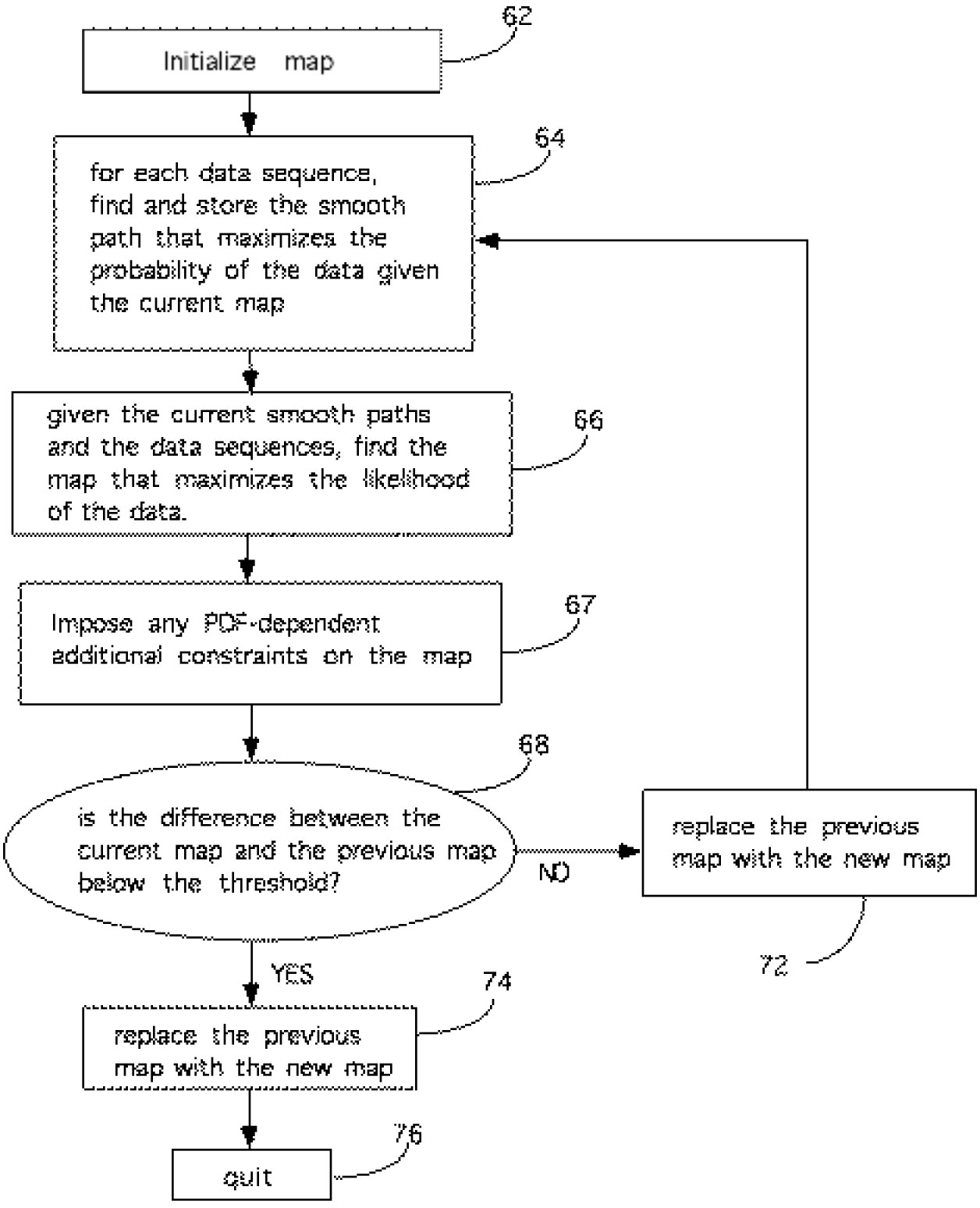
Wheatley, B., Doddington, G., Hemphill, C., Godfrey, J., Holliman, E., McDaniel, J., & Fisher, D. (1992). Robust automatic time alignment of orthographic transcriptions with unconstrained speech. , 533-536.

Zlokarnik, I. (1995). Adding articulatory features to acoustic features for automatic speech recognition. Journal of the Acoustical Society of America, 97(5 pt. 2), 3246(A).

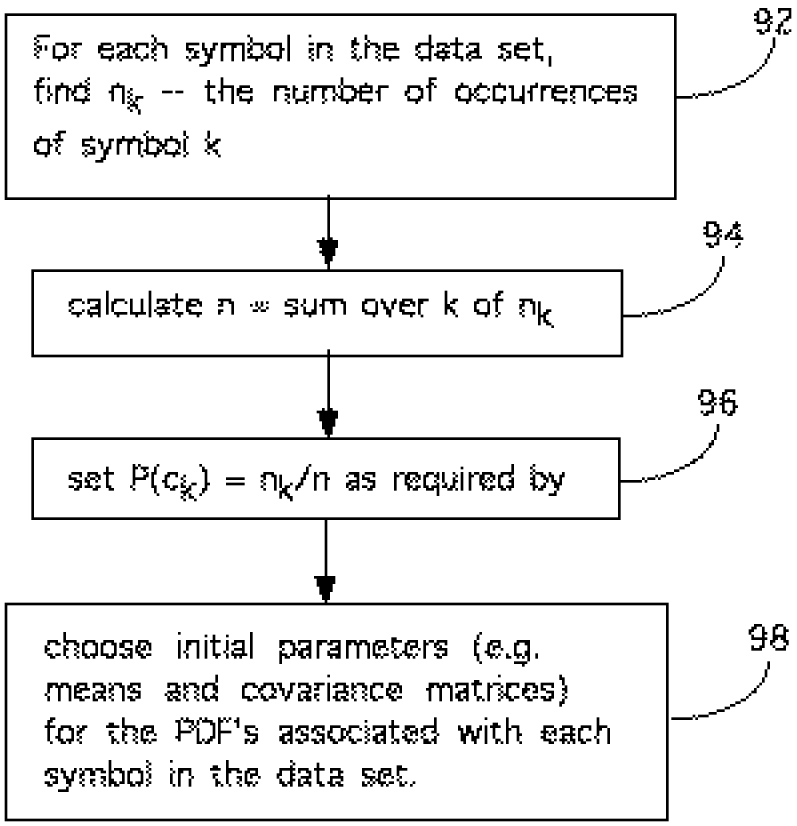
13. Flow Charts



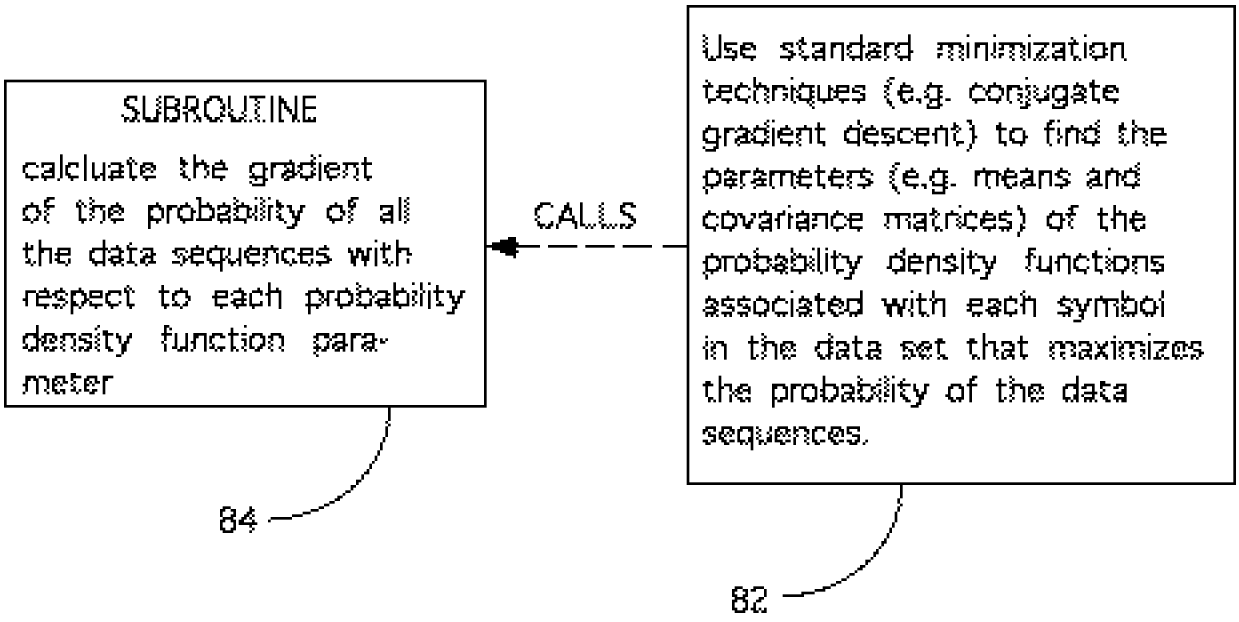
Flow Chart 1


62
Initialize map
64
for each data sequence, find and store the smooth path that maximizes the probability of the data given the current map
66
given the current smooth paths and the data sequences, find the map that maximizes the likelihood of the data.
67
Impose any PDF-dependent additional constraints on the map
68
is the difference between the current map and the previous map below the threshold?
NO
replace the previous map with the new map
72
YES
74
replace the previous map with the new map
76
quit

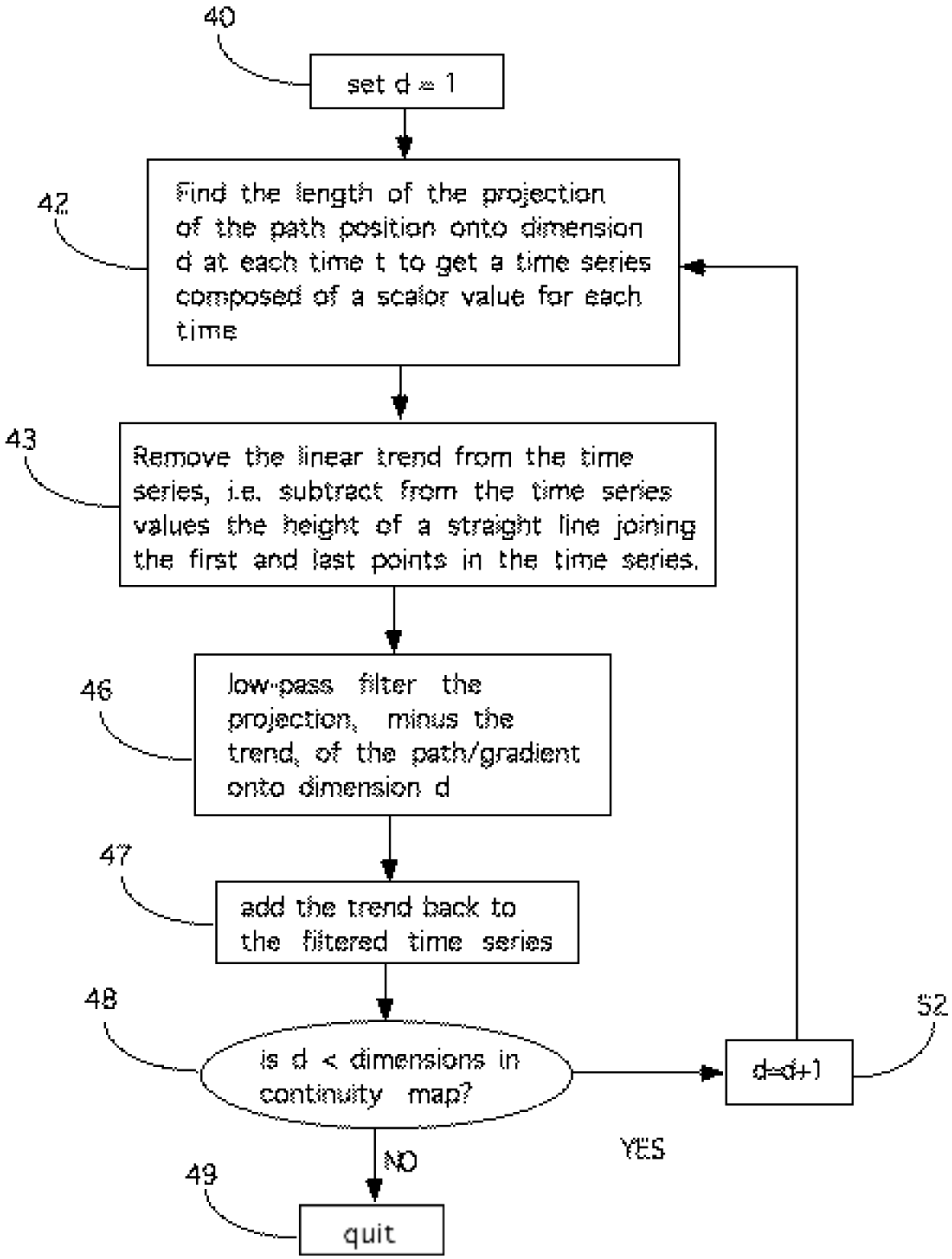
Flow Chart 2


For each symbol in the data set, find n_k -- the number of occurrences of symbol k
92
calculate n = sum over k of n_k
94
set $P(c_k) = n_k/n$ as required by
96
choose initial parameters (e.g. means and covariance matrices) for the PDF's associated with each symbol in the data set.
98

Flow Chart 3


SUBROUTINE
calcluate the gradient of the probability of all the data sequences with respect to each probability density function parameter
CALLS
Use standard minimization techniques (e.g. conjugate gradient descent) to find the parameters (e.g. means and covariance matrices) of the probability density functions associated with each symbol in the data set that maximizes the probability of the data sequences.
84
82

Flow Chart 4


40
set d = 1
42
Find the length of the projection of the path position onto dimension d at each time t to get a time series composed of a scalar value for each time
43
Remove the linear trend from the time series, i.e. subtract from the time series values the height of a straight line joining the first and last points in the time series.
46
low-pass filter the projection, minus the trend, of the path/gradient onto dimension d
47
add the trend back to the filtered time series
48
is d < dimensions in continuity map?
52
d=d+1
YES
NO
49
quit

Flow Chart 5